Vision en mouvement pour la robotique mobile

Matthieu Herrb

Année 1991

THESE

présentée

au Laboratoire d'Automatique et d'Analyse des Systèmes du CNRS

en vue de l'obtention

du Doctorat de l'Université Paul Sabatier de Toulouse

Spécialité : Robotique

par

Matthieu HERRB

Ingénieur ENSEEIHT

VISION EN MOUVEMENT POUR LA ROBOTIQUE MOBILE

Soutenue le 6 février 1990 devant le jury :

G.	GIRALT	Président
M.	BRIOT	Directeur de thèse
S.	CASTAN	Rapporteurs
J.	CROWLEY	Rapporteurs
P.	RIVES	Rapporteurs
A.	CASALS	
M.	DEVY	

Rapport LAAS N° 91048
Cette thèse a été préparée au Laboratoire d'Automatique et d'Analyse des Systèmes du CNRS
7, avenue du Colonel Roche 31 077 Toulouse Cedex

Avant-Propos

Ce mémoire présente la plus grande partie des travaux que j'ai réalisés au Laboratoire d'Automatique et d'Analyse des Systèmes du CNRS durant ces trois dernières années. Je tiens à remercier Monsieur Alain Costes, son directeur, de m'y avoir accueilli.

Je remercie Monsieur Georges Giralt, Directeur de recherches au CNRS pour m'avoir permis de travailler au sein du groupe Robotique et Intelligence Artificielle dont il assure la direction, ainsi que pour avoir accepté de participer à mon jury de thèse.

J'exprime également ma reconnaissance à Monsieur Maurice Briot, professeur à l'Université Paul Sabatier pour avoir accepté d'être mon directeur de thèse.

Merci à Messieurs Serge Castan — professeur à l'Université Paul Sabatier, James Crowley — professeur à l'Institut National Polytechnique de Grenoble et Patrick Rives — chargé recherches à l'INRIA, d'avoir accepter du juger mon travail en étant rapporteurs de cette thèse, et à Madame Alicia Casals –Professeur à l'Université de Barcelone– qui, par sa participation au jury manifeste son intérêt pour mes travaux par dessus les Pyrénnées.

Merci à Raja Chatila et Michel Devy qui ont assuré le suivi scientifique et technique de mes travaux. Leurs conseils et leur grande compétence m'ont été d'une aide précieuse au long de ces trois années.

Gerard Bauzil, Guy Vialaret et Christian Lemaire assurent le fonctionnement des robots de la famille Hilare. Sans leur aide et leur disponiblité, peu de manips auraient été possibles. Merci à eux.

Le travail avec Pierrick, Fabrice et Philippe a été particulièrement enrichissant, tant au niveau scientifique que pratique, et cette collaboration a permis l'intégration de manips complexes avec les robots.

Un grand merci aussi à Marc, Rachid, Danielle pour l'aide qu'ils m'ont apportée au niveau informatique.

Merci enfin au service documentation et édition du LAAS qui a assuré le tirage de cette thèse et plus particulièrement à D. Daurat pour la réalisation du film vidéo des manips.

Sans la présence de Vianney, de Thierry, de Michel, de Victor, de Joël, de Jean-José, de Philippe et des tous les autre collègues et amis, l'ambiance de travail au LAAS et en dehors ne serait pas ce qu'elle est.

Enfin, merci à François qui a réalisé les deux dessins qui égayent cet exposé.

Table des matières

Avant-Propos 1

Table des matières 3

Liste des figures 7

Introduction 9

Présentation des travaux et du mémoire 10

I Vision artificielle et robotique mobile 13

I.1 Le rôle de la perception en robotique mobile autonome 13

I.2 Revue des systèmes existants 15

I.2.1 Les travaux sur les robots 15

I.2.2 Les approches en vision pure 16

I.3 Le projet Hilare 19

II Architectures matérielles pour la vision artificielle 23

II.1 La notion de temps réel 24

II.2 Le traitement d'images parallèle 25

II.2.1 Un modèle pour le traitement d'images parallèle 26

II.2.2 Les architectures 30

II.2.3 La programmation 33

II.3 Les cartes spécialisées Datacube 33

II.3.1 Présentation du système 34

II.3.2 Calcul du gradient sur Datacube 35

II.4 Un processeur général adapté au parallélisme : le Transputer 35

II.4.1 Présentation 35

II.4.2 Les traitements sur réseau de Transputers 38

II.5 Conclusion 39

III Suivi mono-caméra **41**
III.1 Le problème du suivi 41
III.2 Suivi de segments image 42
III.2.1 Représentation d'un segment image 43
III.2.2 Domaine de recherche 46
III.2.3 Extraction des segments correspondants 47
III.2.4 Résultats 47
III.2.5 Discussion 49
III.3 Suivi d'objet 3D 51
III.3.1 Introduction 51
III.3.2 Reconnaissance de l'objet 52
III.3.3 Suivi de l'objet 54
III.3.4 Localisation de l'objet 56
III.3.5 Résultats 61
III.4 Conclusion 63

IV Stéréovision dynamique **65**
IV.1 La stéréovision trinoculaire 67
IV.1.1 La géométrie du système 67
IV.1.2 Les vraies et les fausses contraintes 69
IV.1.3 Prédiction 71
IV.1.4 Vérification 72
IV.1.5 Un sens puis l'autre 73
IV.2 Le suivi des appariements 74
IV.3 Fusion des segments 2D et 3D 75
IV.3.1 Le filtre généralisé pour la manipulation de données imprécises 76
IV.3.2 Modélisation de la stéréovision dynamique 77
IV.3.3 Mise en œuvre 80
IV.4 Résultats 81
IV.5 Conclusion 86

V La vision dans le système de contrôle d'un robot **87**
V.1 Présentation d'Hilare 2 87
V.1.1 La structure informatique 89
V.2 La structure de contrôle d'Hilare 2 91
V.2.1 Les modules fonctionnels 91
V.2.2 Le système de contrôle 94
V.3 Intégration de modules de perception 95

V.3.1 Les fonctions de perception 95
V.3.2 Acquisition de données 98
V.3.3 Contrôle du mouvement par la perception 99
V.3.4 Acquisition de données par stéréovision dynamique 102
V.4 Repères et calibrages 104
V.4.1 Le graphe des repères d'Hilare 104
V.4.2 Calibrage Robot-Platine 105
V.4.3 Calibrage Platine-Caméras 106
V.5 Expérimentations 106
V.6 Conclusion 110

Conclusion générale **111**

A Calcul du gradient sur cartes Datacube **115**
A.1 Calcul du gradient en x et y 115
A.1.1 Limitation de l'amplitude de l'entrée 116
A.1.2 Calcul de la convolution 117
A.2 Tabulation du module et de la phase 118
A.3 Résultats 119

Glossaire **121**

Références bibliographiques **123**

Liste des figures

II.1 Exemple d'architecture décrite par le modèle de Lee et Aggarwal . . 29
II.2 Réseau en grille 4-connexe 31
II.3 Réseau en hypercube de dimension 4 31
II.4 Réseau en pyramide 9-connexe 32
II.5 Architecture interne du Transputer 36

III.1 Le suivi de segments 43
III.2 Les paramètres de la représentation d'un segment de droite 44
III.3 Image initiale de la mire 48
III.4 Image finale de la mire 48
III.5 Image initiale du cube 48
III.6 Image finale du cube 48
III.7 Résultats du suivi 49
III.8 Objet utilisé pour le suivi 53
III.9 L'algorithme de calcul des cliques maximales 55
III.10 Une séquence de suivi de l'objet 61
III.11 Ecart-type (en mètres) de la translation en fonction de la distance et de l'orientation de l'objet 63

IV.1 Le banc de stéréovision d'Hilare 1.5 66
IV.2 Le modèle de projection utilisé pour une caméra 67
IV.3 La contrainte épipolaire 70
IV.4 Schéma de principe de l'algorithme de stéréovision 72
IV.5 Le suivi des appariements de la stéréo 74
IV.6 Le système de stéréovision dynamique 75
IV.7 La scène de bureau utilisée dans le suivi 82
IV.8 Appariements stéréo initiaux de la scène 83
IV.9 L'image 3D originale face) 84
IV.10 L'image 3D après suivi (vue de 3/4 face) 85
IV.11 Nombre de segments suivis lors de chaque itération 85
IV.12 Evolution de la variance des droites 3D au cours de la fusion 85

V.1 Le robot mobile Hilare 2 88
V.2 L'infrastructure informatique embarquée d'Hilare 2 90
V.3 Les interactions entre le planificateur, le contrôle et les modules du robot. 91
V.4 L'architecture du système de contrôle d'exécution 94
V.5 L'architecture du système de perception 97
V.6 Les fonctions du module de suivi d'objet pour la vision 100
V.7 La déclaration de la surveillance sur la détection de l'objet 101
V.8 Le graphe des repères d'Hilare 104
V.9 Tracking avec évitement d'obstacles 107
V.10 Récupération d'erreur au cours du tracking 108
V.11 Suivi du robot Hilare 1 avec coopération multi-robots 109

A.1 Les connexions possibles entre les unités de traitement des cartes . . 116
A.2 Calcul d'une convolution 117
A.3 Calcul tabulé du module et de la phase 118
A.4 Calcul du gradient en pipe-line vrai en utilisant deux cartes VFIR et deux cartes MAX-SP 119

Introduction

Le développement des moyens informatiques dont disposent les chercheurs en robotique rend de plus en plus facile l'utilisation d'algorithmes puissants pour donner aux robots une capacité d'autonomie (le terme d'*intelligence* ne signifiant pas grand'chose pour des machines).

L'autonomie d'un robot est conditionnée par ses capacités de perception : en effet, il n'est pas concevable de réagir face au monde si l'on ne perçoit pas ses éléments. La vision artificielle apparaît comme un moyen priviligié d'acquérir cette perception par la richesse de l'information contenue dans une image, grâce à la disponiblité de la technologie (caméras vidéo de très petite taille, dispositifs de numérisation et de traitement...) et enfin grâce à son caractère anthropomorphique qui séduit naturellement les chercheurs.

Le mouvement est l'un des moyens utilisés par l'homme pour enrichir sa perception visuelle : nous savons faire le tour d'un objet pour le découvrir entièrement, mais nous utilisons aussi le mouvement pour enrichir ou créer une information de profondeur lorsque la vision binoculaire ne produit pas une donnée suffisament précise.

Si l'analyse de séquences d'images préoccupe les chercheurs en vision artificielle depuis un certain nombre d'années, ce n'est qu'assez récemment que la puissance des machines informatiques a permis d'utiliser de manière dynamique l'information fournie par la vision en mouvement dans le système de contrôle d'un robot mobile, pour modéliser son environnement ou guider ses trajectoires.

Les travaux présentés dans ce mémoire ont été réalisés au sein du groupe Robotique et Intelligence Artificielle du Laboratoire d'Automatique et d'Analyse des Systèmes du CNRS. Ils sont intégrés à l'opération Perception et Modélisation

de l'Environnement et rattachés au projet Robots Mobiles. Les robots Hilare 1 et Hilare 1.5 nous ont servi de support expérimental, mais l'ensemble de ces travaux sont surtout destinés à Hilare 2 en cours de développement. Ils constitueront la base de la partie vision de son système de perception.

Présentation des travaux et du mémoire

Cette thèse traite le problème de l'utilisation de la vision artificielle en robotique mobile. Il s'agit d'un domaine où les algorithmes de vision doivent être développés de façon à tenir compte de la dynamique de l'environnement. Pour cela, les travaux ont été menés selon plusieurs axes :

- La vitesse d'exécution des algorithmes : développement d'algorithmes suffisamment rapides pour traiter les images dans des délais compatibles avec un contrôle bouclé sur l'environnement. Cela exige d'une part par l'utilisation de processeurs spécialisés et d'autre part par le recours à des machines parallèles adaptées aux traitements de vision artificielle.

- La prise en compte du mouvement dans les traitements de perception : l'observation de l'évolution des images fournit des données sur la structure de l'environnement, en plus de celles obtenues directement par la perception fixe.

- L'intégration du système de vision artificielle dans le système de contrôle du robot, afin de réaliser la boucle de contrôle : perception — décision — action.

Le chapitre I présente une revue rapide des problèmes qui se posent en robotique dans le domaine de la vision, ainsi que les solutions proposées par d'autres auteurs.

Concernant l'amélioration des temps de calcul nécessaires aux algorithmes de vision, nous avons travaillé sur la vision « bas niveau » : nous avons développé un logiciel d'extraction de segments à partir des images numériques utilisant les cartes de traitement spécialisées Datacube pour le calcul des lignes de gradient puis un réseau de Transputers pour en réaliser une approximation polygonale. Ces travaux sont présentés au chapitre II.

Pour la prise en compte de l'information de mouvement, nous avons réalisé un logiciel de suivi de segments dans une séquence d'images, basé sur une prédiction au premier ordre du mouvement dans les images, suivie d'une vérification qui fait appel à une contrainte externe : appartenance à un objet (dans le cas monoculaire) ou un segment 3D issu de la stéréovision trinoculaire. Il est décrit dans le chapitre III.

Ce logiciel est utilisé dans deux systèmes sur le robot mobile Hilare :

Les mouvements asservis : suivi d'objets tridimensionnels mono-caméra qui permet au robot de localiser un objet connu a priori, mobile ou non, dans l'environnement puis de le rejoindre en maintenant sa caméra orientable pointée vers lui durant son déplacement, tout en évitant d'éventuels obstacles imprévus. Cette application met en lumière quelques-uns des problèmes liés à l'intégration du système de contrôle.

La modélisation de l'environnement : acquisition de données tridimensionnelles robustes par la stéréovision dynamique. Le suivi des segments appariés par un algorithme classique de stéréovision trinoculaire, permet d'enrichir d'une image à l'autre la précision de la localisation tridimensionnelle de ces segments et évite le calcul (long) des appariements à l'arrêt après chaque mouvement. Seuls les segments nouvellement apparus dans le champ des caméras doivent être appariés. (chapitre IV).

Enfin, au chapitre V, nous nous sommes posé le problème de l'intégration de ces systèmes de vision dans l'architecture générale d'un robot mobile. L'analyse des fonctions du robot mobile faisant intervenir la vision (mouvements asservis et modélisation de l'environnement) a permis de définir l'ensemble des modules fonctionnels regroupant les activités de vision, ainsi que les données et les contrôles échangés avec les autres éléments du robot. A titre d'exemple, nous montrons comment le logiciel de suivi d'objets tridimensionnels a été intégré dans le système de contrôle d'Hilare 1.5.

Différentes expérimentations sur le robot mobile Hilare 1.5 ont permis de valider les concepts et les algorithmes proposés.

Chapitre I

Vision artificielle et robotique mobile

Dans ce chapitre nous présentons la problématique générale du sujet qui nous intéresse en fonction des objectifs à atteindre et par rapport aux autres travaux sur des problèmes semblables.

I.1 Le rôle de la perception en robotique mobile autonome

Un robot autonome est un système capable d'effectuer des tâches de navigation ou de manipulation sans intervention humaine directe. Il est conçu pour remplacer l'homme dans des tâches qu'il ne peut ou ne veut faire lui-même, soit parce qu'elles sont trop dures dans le sens *pénible* (cadences de production, environnements hostiles...), soit parce qu'elles sont trop dures dans le sens *difficile* pour lui (dans le cadre sous-marin ou spatial où l'homme ne peut aller, par exemple).

Pour être dotés de capacités de décision autonome, de tels systèmes doivent pouvoir acquérir un modèle de leur environnement, afin de planifier puis d'exécuter leurs missions. De plus ils évoluent dans un monde dynamique où non seulement le (ou les) robot(s) bouge(nt), mais où l'environnement aussi évolue au cours du temps.

A cause de la richesse et de la variété de l'environnement et des situations (du point de vue de la perception au moins), un robot autonome est nécessairement *multicapteurs*. Cela signifie qu'il utilise les données en provenance de plusieurs capteurs à la fois. Ici nous nous intéressons plus particulièrement à la vision artificielle qui utilise une ou plusieurs caméras vidéo comme capteurs.

Globalement, le système de perception doit assurer trois types de fonctions :

- La perception pour le contrôle, destinée à assurer une réactivité immédiate par rapport à l'environnement proche : détection et évitement d'obstacles, suivi de paroi, suivi visuel d'amers... Ces données sont en général fournies par des systèmes de mesure de distance utilisant des capteurs à ultra-sons ou à télémétrie laser.

- Une perception pour la modélisation de l'environnement qui manipule à la fois des données de type géométrique et de type sémantique permettant au système décisionnel d'accomplir les tâches qui lui sont confiées : identification de lieux ou d'objets particuliers tels que portes, postes d'accostage, objets à manipuler ou détection d'amers. Les données sont en général fournies par des caméras ou des systèmes de télémétrie laser.

- La localisation du robot dans cet environnement. Les capteurs proprioceptifs (l'odométrie) fournissent une mesure des déplacements du robot. A cause du caractère cumulatif des erreurs sur cette mesure, il est nécessaire de pouvoir localiser de manière absolue le robot en intégrant une perception courante avec le modèle de l'environnement connu.

  En environnement inconnu, la mise à jour de la position du robot (on parle souvent de recalage) est un «sous-produit» de l'aggrégation des perceptions successives.

En général il n'est pas possible de structurer l'environnement du robot, surtout pour des robots d'intervention ou d'exploration, en plaçant dans son environnement un système de marquage ou de balises. Néanmoins le robot peut disposer de connaissances générales de type géométrique (sol plat, murs verticaux, modèles de portes, de couloirs) ou topologique (décomposition en pièces) sur son environnement.

I.2 Revue des systèmes existants

Les travaux existants dans le domaine de la vision en mouvement pour les robots mobiles sont de deux types complémentaires :

- Les travaux qui intègrent un système de perception dans un robot pour contrôler le mouvement en utilisant les données fournies et en essayant de satisfaire les contraintes de réactivité réelle. Dans ce cas les contraintes de rapidité et de complexité limitent parfois la généralité de l'approche perception.
- Les travaux en vision pure qui essayent, au vu d'un problème donné (estimation du mouvement...), d'apporter une solution dans le cadre le plus général possible en extrayant le maximum d'information des données visuelles.

I.2.1 Les travaux sur les robots

Au MIT[1], Brooks et Horswill [Horswill 88] ont développé un système de guidage pour robots mobiles utilisant la vision ; leur robot est capable de suivre un objet quelconque en mouvement à la cadence de cinq images par seconde en utilisant un système à basse résolution : les images ont 32 lignes de 28 colonnes. Par ailleurs, ce système ne fournit aucune information quantitative sur la position de l'objet. Il indique simplement *à gauche* ou *à droite* ou *en avant* selon la position du *pattern* représentant l'objet dans l'image.

S. Tsuji [Tsuji 88], à l'université d'Osaka, propose un système complet de navigation et de modélisation utilisant la vision. Il utilise la méthode du *Focus Of Expansion*[2] pour estimer le mouvement du robot afin de l'asservir sur une trajectoire ou de calculer la position des éléments suivis (segments de droite verticaux).

Toujours au Japon, Harashima et Kubota [Kubota 88] utilisent la méthode des potentiels dans un système de navigation pour fusionner un but obtenu par la vision et un autre calculé à partir des mesures des capteurs ultra-sonores.

A. Casals et J. Amat développent à l'Université de Barcelone le robot mobile *Eixerit* doté d'importantes capacités de perception multisensorielles : il est doté de

[1] Massachusset Institut of Technology
[2] Voir Glossaire.

divers capteurs (stéréovision, ultra-sons, plan de lumière...) ainsi que d'une architecture informatique spécialisée qui permet un guidage efficace du robot à partir du suivi visuel de balises [Casals 90].

V. Graefe et E. Dickmanns ont réalisé un véhicule qui utilise la vision pour le suivi de route [Dickmanns 90]. Les bords de la route sont suivis par des fenêtres d'intérêt qui permettent d'évaluer par filtrage récursif la courbure et la pente de la chaussée. A partir de cette information les commandes du moteur et de la direction du véhicule sont calculées. L'architecture spécialisée BVV [Graefe 90] permet d'atteindre des vitesses supérieures à 100 km/h sur une autoroute bien dégagée.

Dans le même domaine, le véhicule NavLab de l'université de Carnegie Mellon utilise une caméra laser 3D ERIM en combinaison avec une caméra 2D pour caractériser la partie plane de la chaussée.

En France, outre le projet Hilare dont il est largement question dans ce manuscrit, différents laboratoires français[3] développent le projet ORASIS au sein du Pôle Vision du Programme de Recherche Coordonnée Communication (GRECO-PRC) Homme-Machine.

Enfin, dans le domaine des manipulateurs, F. Chaumette et P. Rives asservissent en boucle fermée un robot manipulateur sur des «signaux visuels» [Chaumette 90]. Il s'agit de points, de cercles ou de droites que l'on souhaite voir dans une certaine configuration. Leur système calcule une commande du bras à partir de l'écart entre la configuration courante et la configuration à atteindre.

I.2.2 Les approches en vision pure

Dans ce cas, les travaux portent sur l'exploitation maximale des données visuelles.

Le flot optique

L'étude de la vision humaine a montré l'existence d'une réaction particulière du système visuel aux mouvements des images sur la rétine: le flot optique. En

[3]Les principaux sont : le CRIN, l'INRIA, l'IRISA, l'IRIT, le LIFIA et l'Université B. Pascal de Clermont-Ferrand

partant de cette constatation, les chercheurs en vision artificielle ont tenté d'établir la relation qui existe entre le mouvement apparent de l'image projetée sur la rétine et le mouvement réel dans l'espace à trois dimensions [Horn 81].

Il est important de noter que le flot optique n'est pas simplement la projection du champ des vitesses de l'espace réel sur l'image. En effet ce qui est perçu, c'est la lumière réfléchie par la scène observée et par conséquent les variations de cette luminance dans le temps — non les vitesses des points de la scène.

Par exemple une sphère de couleur uniforme et sans défauts de surface en rotation sur elle-même ne produit pas de flot optique, alors que le champ des vitesses projeté dans l'image n'est pas nul.

Estimation continue du flot optique

Une des méthodes les plus employées dans l'estimation du champ des vitesses $\vec{\omega}(u, v)$ est la méthode *différentielle*, basée sur un développement au premier ordre de la fonction intensité de l'image, vue comme une fonction spatio-temporelle : $I = f(x, y, t)$. On obtient alors la relation I.1 [Siahmed 86].

$$f_x.u + f_y.v + f_t = 0 \tag{I.1}$$

Cette équation fournit un modèle photométrique local qui permet de calculer la composante (u, v) de la vitesse selon la normale aux contours de l'image.

Cette information n'est pas suffisante pour déterminer l'ensemble du champ des vitesses. On est alors amené à réaliser d'autres hypothèses sur les propriétés du champ recherché, telles que la continuité au premier ordre.

Dans [Nagel 86], on trouve une revue des différentes méthodes utilisées jusque là.

Cette approche présente un certain nombre de limitations bien connues :

- l'équation I.1 repose sur l'hypothèse que la luminance d'un point reste constante, ce qui n'est pas vrai pour les mouvements de rotation.

- dès que l'approximation linéaire de la fonction intensité f perd sa validité, en particulier dans les zones correspondant à des occultations (fortes discontinuités en profondeur) ou des régions où les gradients spatiaux et temporels

n'interagissent pas (zones homogènes ou mouvement trop important), les performances se dégradent.

- les discontinuités du mouvement ne peuvent pas être prises en compte de manière satisfaisante : il n'est pas possible de segmenter une séquence d'images en régions de mouvement uniforme.

Ce dernier inconvénient est traité par F. Heitz et P. Bouthemy qui proposent une approche bayésienne de l'analyse du mouvement utilisant une modélisation par champs markoviens du flot optique. Celle-ci prend bien en compte les régions de forte discontinuité [Heitz 89] et permet de segmenter les séquences d'images en régions de mouvement homogène, à partir de l'étiquetage des champs.

Suivi de caractéristiques

Une autre approche des problèmes de vision dynamique utilise des éléments caractéristiques de l'image (points, droites, régions,...) que l'on suppose liés aux objets de la scène. Il n'est plus nécessaire alors de calculer un champ de vitesse sur toute l'image, et les problèmes liés à l'approximation linéaire de la fonction f évoqués précédemment disparaissent.

Les travaux de Longuet-Higgins sur la détermination du mouvement et de la structure à partir du suivi de points caractéristiques [Longuet-Higgins 81] ont donné naissance à de nombreuses autres études. Son algorithme de calcul de la structure et du mouvement à partir de huit points a été repris et étudié par de nombreux auteurs. M. Demazure a mené une étude mathématique très approfondie sur le nombre et la qualité des solutions en présence de bruit [Demazure 88].

D'autres méthodes ont été développées autour du suivi de points caractéristiques, notamment au LAAS par N. Si Ahmed [Siahmed 86] et H. Garnousset [Garnousset 86] utilisant le *Focus of expansion* pour déterminer séparément les composantes en translation et en rotation du mouvement par des techniques de régulation.

Les points d'intérêt fournissant une information trop dispersée, les travaux ont évolué vers l'utilisation de primitives plus riches, telles que les droites ou les

segments de droite. Deux approches sont utilisées pour la résolution du problème du calcul du mouvement de l'observateur et de la structure de la scène :

- T. S. Huang propose une solution directe basée sur l'estimation aux moindres carrés du mouvement à partir du suivi d'au moins treize droites dans une séquence d'au moins trois images. Il détermine la validité de la solution et peut détecter les cas de dégénérescence.

- O. Faugeras [Faugeras 87a], propose une solution itérative basée sur le filtre de Kalman. Il utilise au minimum six droites dans trois images successives pour calculer une détermination fiable.

La stabilité des solutions dans les images réelles posant des problèmes dans cette approche, un nombre croissant de travaux se proposent de suivre directement des caractéristiques tri-dimensionnelles issues soit de la stéréovision [Faugeras 89] [Crowley 90], soit de caméras laser 3D [Aggarwal 90]. Dans ce cas, la robustesse de la construction provient du fait que l'information de mouvement vient enrichir l'information 3D fournie par chaque image.

I.3 Le projet Hilare

Le robot Hilare sert de support expérimental pour l'étude des problèmes liés à l'autonomie des robots mobiles. Hilare 1 a été construit à partir de 1977, il a été suivi par Hilare 1.5 et puis par Hilare 2, en cours d'intrumentation.

Dès l'origine les travaux menés autour d'Hilare se sont intéressés à tous les aspects de la problématique des robots mobiles : structure décisionnelle, navigation, planification, perception de l'environnement. La structure multi-sensorielle d'Hilare a permis le développement de nombreux travaux sur la structure de perception d'un robot autonome :

- Modélisation incrémentale de l'environnement par télémètre laser 2D et/ou par fusion laser/stéréovision.

- Modélisation par facettes planes grâce à la coopération entre la stéréovision et le télémètre laser 2D.

- Modélisation incrémentale à partir de telles facettes planes.
- Modélisation sur terrain accidenté par télémétrie laser 3D.
- Fusion caméra/télémètre laser 3D
- Reconnaissance d'objets par fusion multicapteurs : stéréovision et/ou caméra couleur, télémètre laser.
- Suivi d'objets.

Ces travaux se trouvent intégrés dans différents projets de recherche nationaux et internationaux dans lesquels Hilare a été impliqué :

Le projet ESPRIT SKIDS a permis durant les trois dernières années la réalisation d'une «machine de perception intelligente» capable de surveiller, modéliser et reconnaître les éléments statiques et dynamiques d'une scène d'intérieur et de planifier et de contrôler l'utilisation des capteurs et des traitements associés en fonction des éléments d'intérêt à observer.

Le projet AMR (Advanced Mobile Robots for Public Safety Applications) a pour objectif d'étudier la réalisation de robots mobiles d'intervention dans des situations d'incidents ou de catastrophes (incendies de toutes origines, incidents de centrales nucléaires, tremblement de terre, etc.) dans des environnements connus devenus hostiles, pour aider à assurer les secours, à examiner ou contrôler la situation.

Le concept AMR consiste en deux unités robotisées : AMR1, véhicule lourd, doté d'un équipement sensoriel et informatique important, transportant sur le site ou à proximité des unités AMRZ, plus légères, et plus autonomes. AMR1 est téléopéré par un opérateur humain, tandis que AMRZ, s'il fonctionne aussi en mode téléopéré, doit être capable d'agir de façon autonome, notamment en cas de coupure des communications. Il dispose d'un équipement embarqué (système de perception multisensoriel incluant une caméra, un télémètre laser et des capteurs à ultra-sons, ainsi que des moyens de traitement, de contrôle, et de communication). Le rôle du groupe RIA du LAAS est l'étude et la réalisation des systèmes temps réel à base de connaissances pour le contrôle autonome d'AMRZ d'une part et la modélisation de l'environnement 3D d'autre part.

Le projet VAP (Véhicule Autonome Planétaire) du CNES, auquel est associé le groupement RISP[4] étudie la réalisation d'un robot d'exploration pour sites planétaires (orienté plus spécialement vers une mission éventuelle sur Mars).

Ce projet comprend plusieurs axes de recherche parmi lesquels nous citons l'étude d'un système de locomotion adapté au type de terrain rencontré, l'étude du système de perception et de modélisation de l'environnement, et le système de contrôle et de planification qui assure l'autonomie et la sécurité du véhicule pour l'exécution de missions téléprogammées.

[4]Robotique d'Intervention sur Site Planétaire : groupement réunissant le CEA, le CNRS, l'INRIA et l'ONERA

Chapitre II

Architectures matérielles pour la vision artificielle

En robotique, comme dans de nombreux domaines, il est important de disposer d'algorithmes rapides. Il est connu qu'en général la vision artificielle est grosse consommatrice de temps de calcul, en raison essentiellement de la taille des données à traiter, ce qui a limité jusqu'ici son développement dans des applications de type robotique.

Les progrès rapides du matériel (intégration de plus en plus poussée des circuits et augmentation de la fréquence des horloges) rendent celui-ci de plus en plus performant et de plus en plus rapide. Par conséquent la vision artificielle voit — comme les autres — ses temps de calcul baisser d'année en année.

Dans ce chapitre nous allons montrer comment des architectures matérielles adaptées peuvent permettre de réaliser de manière quasiment « temps réel » des traitements de base qui nécessitaient il y a peu de temps encore plusieurs secondes de temps CPU. Pour cela nous essaierons d'abord de définir ce que nous comprenons par temps-réel, puis nous présenterons un formalisme adapté à la modélisation des traitements sur architectures parallèles et enfin nous nous intéresserons à la réalisation

d'un système d'extraction de segments de contours réalisé sur une architecture mixte à base de cartes spécialisées Datacube et de Transputers.

II.1 La notion de temps réel

La définition du temps réel est une source de polémique inépuisable. Nous ne prétendons pas ici donner l'ultime définition du terme, mais simplement préciser les concepts nécessaires dans le domaine de la vision artificielle appliquée à la robotique.

- Le temps réel, c'est la possibilité pour des applications d'interagir physiquement avec leur environnement, de réaliser des boucles fermées perception — décision — action ; c'est à dire que les temps de calcul sont compatibles avec la vitesse d'évolution des paramètres extérieurs du système que l'on veut contrôler (par exemple : le suivi d'un objet mobile).

- Le temps réel, c'est des temps de calcul compatibles avec une exécution performante de la tâche. Il s'agit ici d'un critère subjectif qui dépend fortement de l'application : pour un assemblage de pièces industrielles il s'agit de cadences supérieures à ce qu'un ouvrier humain peut faire, alors que dans des applications de type sous-marin ou spatial, le temps importe moins actuellement.

- Le temps réel, c'est des temps d'exécution bornés ou connus à priori, qui permettent à un système de planification et de contrôle de raisonner sur la tâche en intégrant la notion du temps.

- Le temps réel, c'est la capacité pour un système de détecter des perturbations asynchrones et d'y réagir en un temps compatible avec l'évolution de l'environnement (c'est-à-dire les cadences des perturbations) et de la tâche.

De plus, en vision artificielle, le standard vidéo RS-170 introduit une contrainte forte sur la notion de temps réel : 25 images par seconde. Cette cadence est celle à laquelle les images peuvent être acquises par l'immense majorité des caméras disponibles. C'est beaucoup si l'on souhaite traiter toutes les images, mais c'est peu si on souhaite suivre des phénomènes plus rapides. Signalons dans ces cas-là la possibilité d'acquérir des informations à la fréquence « pixel » d'une caméra CCD.

II.2 Le traitement d'images parallèle

De nombreuses sociétés ou laboratoires ont développé des systèmes matériels destinés au traitement et à l'interprétation d'images. On peut les classer en trois catégories[1] :

Systèmes dédiés : il s'agit de développements très spécialisés de systèmes fermés destinés à traiter un seul type d'application, généralement dans le domaine industriel (par exemple le *Visiomat* de la société Robotronics dont l'extracteur de contours a été développé au LAAS).

Systèmes spécialisés : il s'agit de systèmes de cartes spécialisées pour le traitement d'images, mais dotées de possibilités d'adaptation importantes par programmation ou par configuration matérielle. Les cartes *Imaging*, *Matrox* ou *Datacube* rentrent dans cette catégorie, ainsi que les differents éléments développés dans le cadre du projet Esprit P 940 (extracteur de contours, token-tracker, stereo-matcher).

Systèmes généraux : il ne s'agit pas d'une architecture spécialement développée pour le traitement d'images. Les calculs sont faits sur une machine séquentielle (processeur de traitement du signal – DSP) ou parallèle (par exemple un réseau de transputers) générale dont la puissance est adaptée aux traitements à réaliser. Ces systèmes se distinguent des calculateurs à usage général par la présence de dispositifs d'acquisition et de visualisation des images.

Les systèmes développés dans les laboratoires se classent essentiellement parmi les deux dernières catégories.

Les modules spécialisés sont en général utilisés pour les traitements de bas et moyen niveaux : Extraction de contours [Ranganathan 88], suivi de lignes [Chehikian 88],...

Néanmoins il existe des systèmes destinés aux traitements de bas niveau développés sur des processeurs généraux : British Aerospace a développé un ex-

[1] En raison de l'évolution très rapide du matériel, nous ne citons que les réalisations récentes

tracteur de contours basé sur l'algorithme de Canny sur un réseau de Transputers [Sheen 88] ; à l'université de Carnegie Mellon divers algorithmes de bas niveau ont été implémentés sur le processeur *WARP* [Hamey 87].

Les traitements dits de haut-niveau sont plus facilement implémentés sur des systèmes à base de processeurs généraux : *iWARP* [Bockar 89], *OPSILA* [Duclos 88].

II.2.1 Un modèle pour le traitement d'images parallèle

Dans [Lee 90], Lee et Aggarwal proposent un formalisme pour la modélisation de systèmes de traitement d'images de type parallèle.

Le traitement d'images est un domaine où le parallélisme peut intervenir à plusieurs niveaux. Lee et Aggarwal distinguent le parallélisme au niveau *image* – ou parallélisme spatial et le parallélisme au niveau *fonction* – ou parallélisme temporel.

Le parallélisme au niveau image utilise le fait qu'un même traitement est appliqué à toute l'image, soit pixel par pixel, soit région par région. Les données d'entrée et de sortie peuvent alors être partitionnées en sous-domaines qui seront traités en parallèle, chacun sur un processeur différent. Il faut noter qu'il peut y avoir des recouvrements entre les sous-domaines, par exemple pour appliquer des fonctions utilisant un voisinage de chaque pixel.

Le parallélisme au niveau fonctionnel correspond au fait qu'un traitement est en général composé d'une chaîne d'opérations élémentaires qui sont exécutées de manière séquentielle. Dans les applications de type temps réel ou robotique, où l'on traite des séquences d'images, il est intéressant d'enchaîner les opérations selon un *pipe-line* : dès que la première image a franchi la première opération élémentaire, le traitement de l'image suivante peut commencer.

Considérons maintenant un traitement d'images représenté de la manière suivante :

$$O = f(I) \tag{II.1}$$

f est un opérateur de traitement d'images (par exemple l'extraction de segments de contour), I est l'ensemble des données d'entrée (l'image initiale) et O l'ensemble des résultats (par exemple la liste des segments extraits).

Le modèle proposé par Lee et Aggarwal partitionne f et/ou I pour exploiter les deux types de parallélisme et effectuer le maximum de traitements en parallèle selon le paradigme *diviser pour régner*.

Le parallélisme spatial

La partition de l'entrée I en plusieurs sous-domaines permet de réécrire l'équation II.1 sous la forme :

$$\begin{aligned} O &= f^1(I_1) \cup f^2(I_2) \cup \ldots \cup f^M(I_M) \\ &= \bigcup_{i=1}^{M} f^i(I_i) \end{aligned} \tag{II.2}$$

$\{I_i\}$ est l'ensemble des sous-domaines de I. f^i est la restriction de f à I_i, M le nombre de sous-domaines et $\cup$ représente l'opération réalisée pour combiner les résultats et dépend de la fonction. En particulier cette opération peut être effectuée directement durant le calcul des f^i ou nécessiter une étape explicite.

Les I_i ne sont pas forcément tous disjoints ; dans le cas d'une opération de convolution, les domaines I_i doivent être plus larges que la partie d'image à calculer, avec des recouvrements. D'autres types de découpage des données peuvent également être pris en compte : dans le cas de mise en correspondance de modèles avec une image les modèles peuvent être répartis sur les différents processeurs : chaque processeur recherchant alors le(s) modèle(s) qu'il connaît dans l'image qui lui est présentée.

A priori, le nombre d'éléments dans la partition de I peut être élevé (on peut envisager jusqu'à un processeur par pixel), et on peut donc espérer gagner plusieurs ordres de grandeur en augmentant M.

Néanmoins, au fur et à mesure que M augmente, le temps de calcul de f^i décroît et donc les temps de communication deviennent prépondérants. L'efficacité globale stagne ou même décroît après une certaine valeur de M, si le temps de communication croît avec M (le nombre de processeurs). Cette efficacité globale dépend du type d'architecture et du type d'algorithme exécuté.

Le parallélisme fonctionnel

De manière similaire la décomposition en opérations élémentaires de f permet de l'écrire comme composition de plusieurs fonctions correspondant aux opérations successives. On peut donc réécrire II.1 de la manière suivante :

$$\begin{aligned} O &= f_k \circ f_{k-1} \circ \cdots \circ f_2 \circ f_1(I) \\ &= (\prod_{j=1}^{K} f_j)(I) \end{aligned} \tag{II.3}$$

K est le nombre d'opérations et $\prod$ représente la composition de fonctions.

Contrairement au cas du parallélisme spatial, le nombre K des opérations élémentaires enchaînées n'est pas forcément très élevé.

Si l'on considère que l'on traite des séquences suffisamment longues pour négliger le temps de chargement initial du pipe-line, le temps de traitement d'une image est le temps de traitement de l'étage le plus lent. Il est donc intéressant de prévoir le découpage des f_j de manière à diminuer ce temps et à garder les autres temps de traitement proches de cette valeur maximale pour minimiser le coût de l'ensemble et maximiser le taux d'occupation des processeurs.

Le modèle complet

En combinant les deux types de parallélisme (II.2 et II.3) on peut donc réécrire II.1 :

$$\begin{aligned} O &= f(I) \\ &= \prod_{j=1}^{K} \left(\bigcup_{i=1}^{M_j} f_i^j(I_{ij}) \right) \end{aligned} \tag{II.4}$$

L'équation II.4 correspond à un pipe-line composé de K étages, qui réalisent chacun un traitement en utilisant M_j processeurs, tel que représenté sur la figure II.1.

Ce modèle suppose implicitement que tous les processeurs d'un étage du pipeline aient terminé leurs calculs avant que l'étage suivant ne commence les siens.

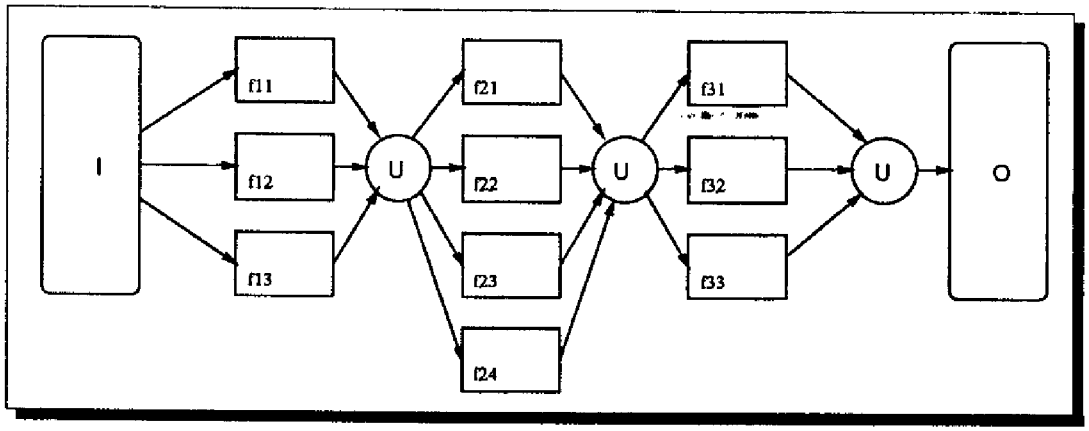


Figure II.1: *Exemple d'architecture décrite par le modèle de Lee et Aggarwal*

Dans le cas du traitement de séquences d'images cette contrainte permet de borner le temps de transit dans le pipeline d'une trame.

Le temps de traitement d'un tel système dépend de trois types de paramètres :

- le temps de calcul d'une fonction f_j^i (calcul de la j-ième partition à l'étage i) ; nous le notons T_{cij}.
- le temps de communication entre les processeurs à l'intérieur de l'étage i (essentiellement le temps nécessaire au calcul de la fonction $\cup$). Il dépend de la structure interne de l'étage i (volume de communication, topologie,...) ; nous le notons T_{ei}.
- le temps de communication entre l'étage i et l'étage $i+1$ (que l'on peut voir comme le temps de la fonction $\prod$). Il dépend de la structure du pipe-line ; nous le notons T_{pi}.

Les durées de traitement des différents processeurs d'un étage du pipe-line ne sont pas nécessairement égales. La durée globale pour l'étage i est :

$$T_{ci} = \max_j(T_{cij})$$

Le temps de cycle complet d'un étage du pipe-line est donc donné par :

$$t_i = T_{ci} + T_{ei} + T_{pi}$$

Par conséquent le temps de cycle global du pipe-line est :

$$t_0 = \max_i(t_i)$$

Lee et Aggarwal complètent cette étude en la généralisant au cas «dynamique» où le temps de calcul d'un processeur élémentaire n'est pas constant, mais est une variable aléatoire ; ils utilisent alors les modèles probabilistes de files d'attente pour calculer le débit de l'ensemble.

Cette décomposition en parallélisme spatial et fonctionnel est à distinguer de la séparation classique SIMD[2]/MIMD[3] : en effet un étage du pipe-line n'est pas forcément de type SIMD, plusieurs fonctions différentes peuvent être calculées sur des sous-domaines différents.

II.2.2 Les architectures

Nous avons vu que la performance d'un système dépend de la puissance de chaque processeur élémentaire, mais aussi pour une grande part du temps de communication entre les processeurs, soit à l'intérieur d'un étage du pipe-line, soit entre deux étages.

Il existe deux solutions pour la connexion entre les processeurs :

- liaisons par canal de communication (échange de messages) — point à point ou multi-points.
- liaisons par mémoire commune (en général entre 2 processeurs)

Dans le cas de liaisons multi-point par canal de communication (type bus), le problème de la topologie du réseau ne se pose pas en général, mais apparaît un problème d'allocation de la ressource «bus».

Dans le cas des liaisons point à point, par canal de communication ou par mémoire commune, la topologie du réseau influe beaucoup sur les performances du

[2] *Single Instruction Multiple Data :* Une seule instruction exécuté en même temps sur plusieurs données

[3] *Multiple Instruction Multiple Data :* Plusieurs instructions en même temps sur des données différentes

système de communication. La topologie est définie par le nombre de liens disponibles pour chaque processeur élémentaire et la structure des liaisons réalisées.

On trouve dans les systèmes existants divers types de topologies [Fountain 86] :

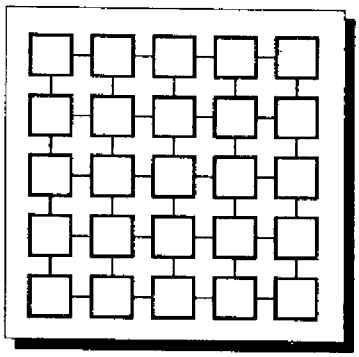


Figure II.2: *Réseau en grille 4-connexe*

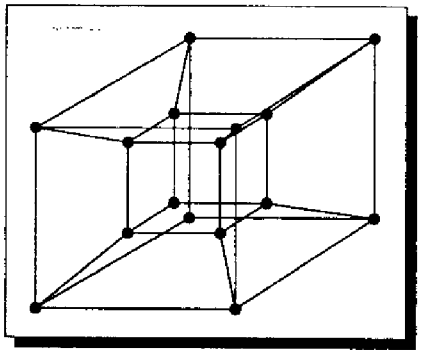


Figure II.3: *Réseau en hypercube de dimension 4*

les grilles : (figure II.2) le réseau est plan, et chaque processeur élémentaire est connecté à un certain nombre (4, 6 ou 8 en général) de ses voisins. Ce type de connexion est très utilisé pour les applications qui nécessitent une communication locale (propagation de contraintes...).

Les processeurs de la famille CLIP [Duff 88] utilisent une structure de grille 4-connexe.

les hypercubes : (figure II.3) dans un réseau à structure hypercube, les 2^n processeurs sont placés sur les sommets d'un cube en dimension n. Chaque arête du cube est un lien entre deux processeurs. Le nombre de connexités N d'un processeur est égal à n. Cette architecture minimise la longueur du chemin entre deux sommets quelconques du réseau, et est donc très utile dans les applications où les communications peuvent se faire de manière aléatoire entre processeurs. La *Connexion Machine* utilise cette architecture.[Little 87].

les pyramides : (figure II.4) dans une structure de type pyramidal, les processeurs élémentaires sont organisés hiérarchiquement en arbre, avec des

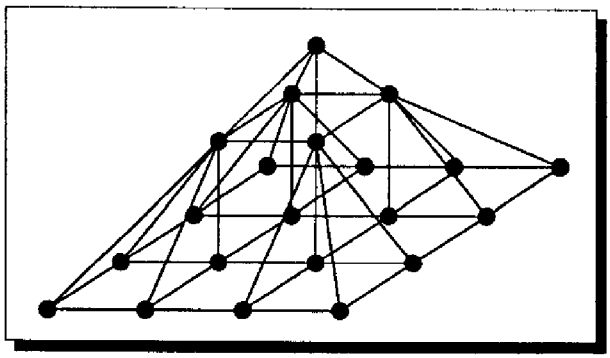


Figure II.4: *Réseau en pyramide 9-connexe*

connexions de type grille entre les processeurs d'un même niveau. Cette topologie est bien adaptée aux opérations du type classifications, histogrammes, séparation/fusion de régions...

La machine SPHINX [Merigot 89] réalisée conjointement par l'IEF[4] et l'ETCA est une machine de type multi-SIMD : chaque étage de la pyramide fonctionne en mode SIMD, les différents étages formant un pipe-line.

Meer [Meer 89] et Jolion [Jolion 89] utilisent des pyramides adaptatives dont la structure épouse la structure de régions de l'image.

les processeurs ligne : ils sont constitués d'un ensemble de processeurs élémentaires connectés en ligne, capables de traiter une ligne (ou une colonne) d'un image en parallèle. Le processeur SYMPATI 2 développé coinjointement par l'IRIT et le CEA peut être constitué de 32 à 128 processeurs, fonctionnant en mode SIMD [Juvin 88][Basille 90]. Sa structure particulière (helicoïdale) permet également d'accéder de manière rapide aux voisins d'un pixel ou de l'utiliser pour des application de haut niveau, par exemple la stéréovision.

les crossbar programmables : certains systèmes disposent maintenant de mediums de communication programmables : les liens de communication des processeurs élémentaires sont connectés à un dispositif matriciel programmable qui permet de réaliser n'importe quelle connexion point à point. Ces systèmes sont soit statiques (La configuration se fait à l'initialisation

[4]Institut d'Electronique Fondamentale, Université de Paris Sud

des processeurs élémentaires) soit dynamiques (La reconfiguration peut intervenir durant l'exécution d'un programme, par exemple entre deux étages du pipe-line fonctionnel). Le *Super Node* de la société Telmat est une machine à crossbar programmable statique.

II.2.3 La programmation

Pour programmer ces nouveaux systèmes, plusieurs approches sont possibles :

- Accès aux registres de contrôle des processeurs par des fonctions de bibliothèque. (En général c'est l'approche utilisée pour les systèmes spécialisés ; citons notamment la bibliothèque *MaxWare* pour les cartes Datacube).
- L'extension d'un langage existant par l'adjonction de mécanismes adaptés (C avec *fils* d'exécution[5], Fortran vectoriel).
- Le développement de nouveaux langages et environnements de programmation adaptés au processeurs (par exemple OCCAM sur les Transputers ou APPLY sur machine WARP [Hamey 89]).
- Certaines architectures se prêtent à l'utilisation de nouveaux modèles pour la programmation du type data-flow [Zavidovique 88].

II.3 Les cartes spécialisées Datacube

Les cartes de la famille *MaxVideo* forment une famille de cartes de traitement d'images en temps réel vidéo conçues pour offrir une grande flexibilité de programmation. Elles sont construites autour de deux bus : le bus VME qui permet le dialogue des cartes avec le système hôte et le bus vidéo spécialisé *MaxBus* qui assure les transferts des images entre les cartes.

Il est possible de réaliser une grande variété d'architectures de traitement d'images avec ces cartes. Les traitements peuvent être organisés en parallèle ou en

[5]Voir glossaire.

pipe-line grâce à la flexibilité de MaxBus. Par construction le temps de cycle d'un étage de pipe-line est égal au temps de trame du signal vidéo.

Ce système de cartes est très populaire dans la communauté de la vision artificielle.

II.3.1 Présentation du système

Nous présentons ici les différents élément du système MaxVidéo que nous avons utilisés pour nos traitements.

La carte d'acquisition et de visualisation

La carte d'acquisition de la gamme MaxVideo que nous utilisons est la *DIGIMAX*. Il s'agit d'une carte de numérisation au standard RS-170 qui fournit à partir de chaque trame vidéo une image 512x512x8bits.

La carte permet de régler le gain du circuit convertisseur analogique/numérique, d'ajouter un signal constant (*offset*) et d'intercaler un filtre sur le signal analogique.

L'affichage des images se fait par l'intermédiaire de trois convertisseurs numérique/analogique pour les canaux rouge, vert et bleu, associés à trois tables de transcodage permettant la visualisation en fausse couleur.

Les cartes mémoire image

Nous utilisons deux types de cartes mémoire image :

FRAMESTORE est une carte relativement simple qui dispose de trois mémoires image de 512 x 512 octets adressables séparément.

ROISTORE est une carte plus sophistiquée qui dispose de deux plans d'un mega-octet chacun, découpables par l'utilisateur (par exemple en huit images de 512 x 512) et accessibles directement dans l'espace d'adressage de la tâche.

De plus cette carte permet de programmer des traitements sur des régions d'intérêt de l'image (ROI) en produisant les signaux de synchronisation

correspondants sur MaxBus, ce qui permet alors de dépasser la cadence vidéo (données moins volumineuses).

La carte Convolueur temps réel

La carte *VFIR* est une carte qui permet de calculer soit une convolution bi-dimensionnelle 3x3, soit une convolution linéaire de dix points de large sur toute l'image en temps réel vidéo (40ms/image).

La carte Unité arithmétique image

La carte *MAX-SP* est une unité de traitement d'images d'usage général. Elle calcule en temps réel vidéo des filtres mono-point dans le domaine spatial ou temporel, la fusion de plusieurs images, la soustraction ou l'addition d'images, le minimum ou le maximum et bien d'autres fonctions, elle contient une LUT de grande capacité qui permet de tabuler le résultat d'une fonction.

II.3.2 Calcul du gradient sur Datacube

Nous avons implémenté le calcul du module et de la phase du gradient sur un ensemble de cartes Datacube. La description complète de cette implémentation se trouve en annexe A.

II.4 Un processeur général adapté au parallélisme : le Transputer

Les Transputers sont des processeurs d'usage général dotés de caractéristiques qui les destinent aux traitements parallèles multi-processeur.

II.4.1 Présentation

Le Transputer

Le Transputer est construit autour d'un processeur 32 bits dont l'architecture est de type RISC[6] — Processeur à jeu d'instructions réduit, et intègre une

[6] *Reduced Instruction Set Computer*

mémoire locale et une unité de calcul flottant. Son jeu d'instructions est constitué d'un noyau d'instructions très rapides et très simples permettant d'avoir des codes très compacts, notamment en calcul flottant. Le Transputer est conçu pour exécuter de façon optimale les traitements parallèles et peut donc traiter en temps partagé un nombre quelconque de processus.

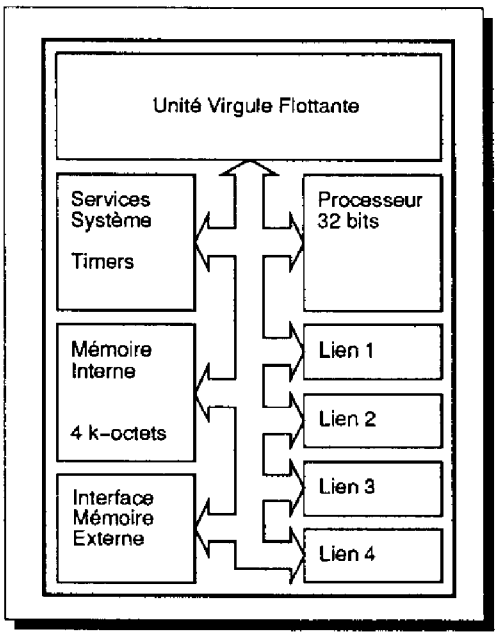


Figure II.5: *Architecture interne du Transputer*

Un Transputer est un processeur doté de:

- une unité arithmétique virgule flottante
- une mémoire locale de quatre kilo-octets et un accès à une mémoire externe de taille quelconque.
- quatre canaux de communication bidirectionnels à haut débit

Ces quatre liens de communication bi-directionnels (jusqu'à 20 mega-bits par seconde) permettent la réalisation de réseaux en assurant les transferts inter-processeurs à haut débit.

Par ailleurs, c'est un circuit très facile à mettre en œuvre, ce qui permet son utilisation dans des systèmes embarqués.

Utilisation d'un réseau

Les canaux de communication du Transputer le destinent à la réalisation de machines multi-processeurs à mémoire locale. En effet, il n'existe aucun dispositif destiné à gérer une mémoire commune sur le Transputer.

Le nombre réduit de canaux (quatre) contraint par ailleurs la topologie des réseaux qui peuvent être construits.

Le concept de «ferme de processeurs»

Le langage C parallèle propose un modèle simple de parallélisation pour les applications de type SIMD (*Single Instruction, Multiple Data*) ou SPMD (*Single Program, Multiple Data*) : la «ferme de processeurs». Les données sont découpées en paquets de taille prédéfinie et envoyées sur le réseau pour traitement. Chaque Transputer du réseau exécute une tâche de traitement puis renvoie les résultats à la racine.

Le routage des paquets de données est assuré en parallèle du traitement sur chaque processeur selon un chemin linéaire du type *Pipe-line*.

Ce modèle n'est pas optimal, notamment à cause du routage qui n'emprunte pas toujours le chemin le plus court. Son efficacité dépend du rapport entre le temps nécessaire au routage d'un paquet de données et le temps de traitement de ces données. Si le temps de routage domine, l'efficacité est bornée par le débit du réseau.

Problèmes et limitations

Pour les applications de type traitement d'images, l'absence de mémoire commune constitue un lourd handicap. En effet, cela contraint les données de fort volume à transiter par les canaux de communication. A 20 Mbits/seconde (le débit actuel des liens INMOS) le transfert d'une image 512 x 512 nécessite au moins 100 millisecondes, soit presque trois trames vidéo.

Néanmoins il existe des cartes de type «Transputer framestore» [Sheen 88] qui permettent — grâce à une mémoire double accès — d'acquérir directement l'image numérique dans la mémoire d'un ou plusieurs Transputers du réseau, en supprimant ou en réduisant ainsi le temps de chargement des données.

II.4.2 Les traitements sur réseau de Transputers

Nous disposons d'une machine *T-node* de la société Telmat composée de trente deux Transputers disposant chacun de 32 kilo-octets de mémoire locale. Les processeurs sont reliés à un réseau de communication du type crossbar programmable qui permet de réaliser de nombreuses configurations.

La programmation se fait en langage « C » et nous utilisons actuellement le modèle de la ferme de processeurs pour exploiter le parallélisme.

Elimination des non-maxima locaux

L'algorithme d'amincissement des lignes de gradient de Canny [Canny 83] a été implémenté sur Transputers en utilisant le concept de ferme de processeurs présenté ci-dessus.

L'image du module du gradient calculée par Datacube est découpée en sous-images (une sous-image par processeur). Comme le traitement n'est pas très important, le temps de routage domine largement le temps de traitement.

Chaînage des points de contour

Le chaînage des points de contour est réalisé à la suite de l'amincissement des lignes à l'intérieur de chaque image par un algorithme de seuillage adaptatif. Lors de la réception des chaînes par le Transputer maître, les connexions entre les sous-images sont réalisées.

Approximation polygonale

L'approximation polygonale est réalisée par un nouveau flot de calcul sur la ferme de processeurs : chaque chaîne est envoyée sur un processeur qui calcule son approximation polygonale et renvoie la liste de segments extraits.

Résultats

1. Comparaison Transputer/mc68020

   Le tableau ci-dessous donne le temps d'exécution de l'amincicement des lignes

de gradient obtenu pour un seul Transputer, comparé à celui obtenu sur un 68 020 dans une station Sun 3/60.

Transputer	68 020
1.1 sec	0.62 sec

Le Transputer seul est donc environ deux fois moins rapide qu'un 68020 pour ce genre d'application (pas de calcul flottant, nombreux accès mémoire).

2. **Efficacité du parallélisme**

   Trois facteurs entrent en jeu : le nombre de processeurs élémentaires, le volume des données et le temps de calcul pour chaque processeur. En effet, dans le modèle de la ferme de processeurs, on ne peut pas intervenir sur les temps de communication : ceux-ci sont fixés par le nombre de processeurs et le volume des données.

   Comme on peut s'y attendre, le meilleur résultat en termes d'efficacité du parallélisme est obtenu avec un faible volume de données et de grands temps de calcul sur chaque processeur élémentaire.

II.5 Conclusion

Dans ce chapitre nous avons constaté que les besoins de la vision artificielle appliquée à la robotique (avec des contraintes de type temps réel) en moyens de calcul conduisent les chercheurs à s'intéresser aux deux aspects des nouvelles architectures informatiques : les matériels spécialisés et les processeurs parallèles. Un formalisme général de modélisation des architectures de ce type a été présenté.

L'implémentation sur processeurs spécialisés Datacube et sur réseau de Transputers des maillons de la chaîne d'extraction des segments de contour des images laisse entrevoir la possibilité de pouvoir réaliser d'ici peu cette opération «de base» à une cadence compatible avec le temps réel vidéo.

Par ailleurs, les facilités croissantes offertes aux développeurs de processeurs spécialisés rendent ceux-ci de plus en plus programmables, et par conséquent la frontière entre les processeurs spécialisés et généraux tend à s'estomper.

Chapitre III

Suivi mono-caméra

Dans ce chapitre, nous présentons le principe du suivi de segments dans une séquence d'images puis une application directe: le suivi d'objet 3D mono-caméra utilisé sur Hilare pour réaliser des mouvements asservis sur une cible visuelle. Une deuxième application du suivi de segments : la stéréovision dynamique sera présentée dans le chapitre IV.

III.1 Le problème du suivi

La vision dynamique, ou vision en mouvement, trouve de nombreuses applications en robotique. Celles-ci peuvent être classées de différentes manières.

Tout d'abord, on peut faire une classification selon la *finalité* du processus de vision à l'intérieur du système :

- La détermination du mouvement, soit de l'observateur, soit d'un objet de l'environnement
- La détermination de la structure de l'environnement.
- L'asservissement du système sur la base de caractéristiques visuelles.

De nombreux systèmes réalisent plusieurs objectifs en même temps, par exemple : structure et mouvement, ou mouvement et asservissement.

Globalement, tous les processus de vision dits «dynamiques» font intervenir une forme de suivi, afin d'évaluer le mouvement apparent dans l'image.

- Le suivi global (dense) de tous les points de l'image. Le problème posé est alors celui de l'estimation du flot optique, c'est à dire de la dérivée temporelle de la fonction image.
- Le suivi d'éléments particuliers dans l'image :
  - Points d'intérêt (selon des critères de luminosité, de courbure locale des contours,...) Dans ce cas, le flot optique peut être calculé pour les points suivis [Siahmed 86].
  - portions de contour (segments de droites ou de courbe). Dans ce cas l'information de flot optique ne peut pas toujours être récupérée entièrement, à défaut de correspondance explicite point à point. (Occultations sur les extrémités de contour).
  - Balises naturelles ou artificielles (bords de routes, murs,...) correspondant à des motifs particuliers dans l'image [Matthies 89].

III.2 Suivi de segments image

Le problème qui est posé ici est à la base des différentes applications du suivi de segments présentées par la suite :

Etant donné un segment de contour dans une image, trouver une liste de segments correspondants possibles dans une nouvelle image.

La solution proposée repose sur deux hypothèses :

- Une prédiction de la position du nouveau segment peut être faite uniquement sur la base des données image bi-dimensionnelles grâce à la continuité du mouvement en trois dimensions, qui induit la continuité du mouvement apparent dans l'image.

- En cas d'ambiguïtés, on fournit l'ensemble des correspondants potentiels trouvés. D'autres contraintes (stéréovision, appartenance à un objet rigide...) seront utilisées pour résoudre l'ambiguïté.

  L'utilisation de cette contrainte extérieure supplémentaire permet de garder une cohérence globale des appariements du suivi à un coût algorithmique relativement faible.

Le schéma de suivi utilisé est présenté dans la figure III.1 : à partir du modèle de chaque segment image, une prédiction sur sa position dans la nouvelle image est calculée. On utilise ensuite un critère de ressemblance pour choisir parmi les segments une liste de candidats. Les contraintes externes permettent alors de lever les ambiguïtés. A partir de la nouvelle observation on peut ensuite mettre à jour le modèle. C'est un algorithme de type *prédiction-vérification.*

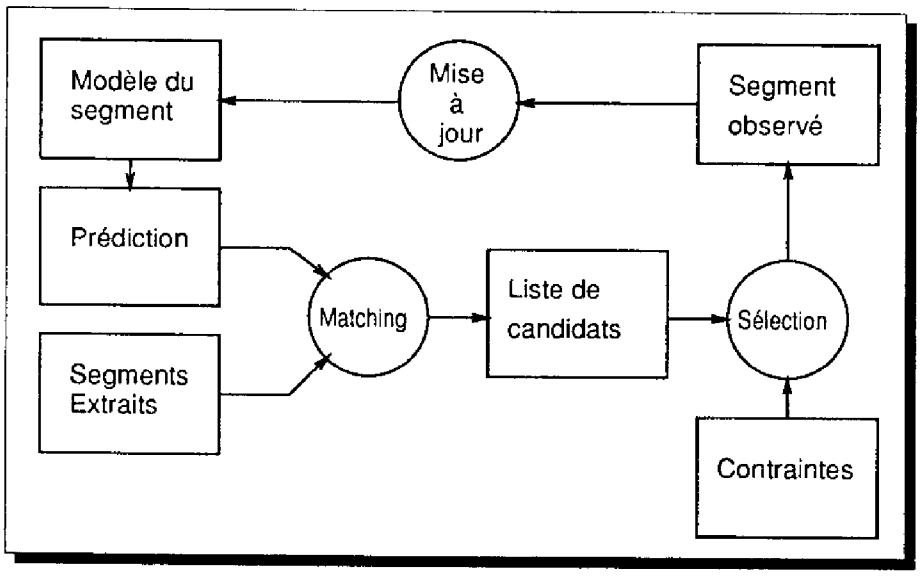


Figure III.1: *Le suivi de segments*

III.2.1 Représentation d'un segment image

Un segment image S est représenté par 4 paramètres. Plusieurs représentations sont possibles (figure III.2) :

- Le vecteur $(x_1, y_1, x_2, y_2)^T$ des coordonnées des 2 extrémités du segment $M_1(x_1, y_1)$ et $M_2(x_2, y_2)$.

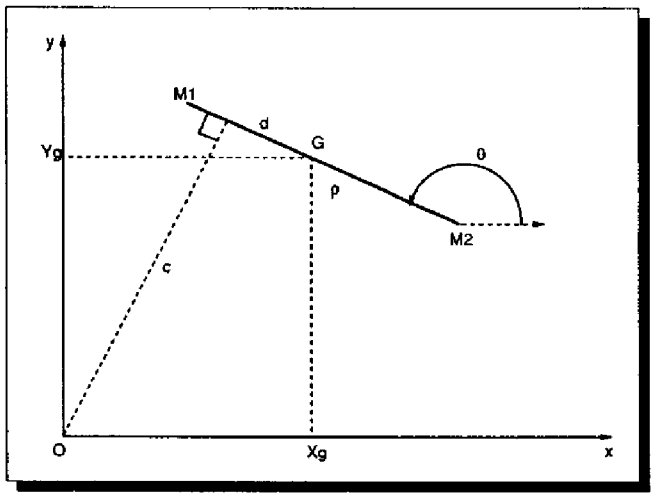


Figure III.2: *Les paramètres de la représentation d'un segment de droite*

- Le vecteur $(x_G, y_G, \theta, \rho)^T$ composé des coordonnées du centre de gravité $G(x_G, y_G)$, de l'orientation θ et la longueur ρ.
- Le vecteur $(c, d, \theta, \rho)^T$ où c est la distance de l'origine à la droite support de S et d la distance entre le milieu de S et la projection de l'origine sur sa droite support [Pham 89].

Les différentes représentations sont liées par les équations suivantes :

$$x_G = \frac{x_1 + x_2}{2} \tag{III.1}$$

$$y_G = \frac{y_1 + y_2}{2} \tag{III.2}$$

$$\theta = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \tag{III.3}$$

$$\rho = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{III.4}$$

$$c = \frac{x_2 y_1 - x1 y_2}{\rho} \tag{III.5}$$

$$d = \frac{(x_2 - x_1)(x_1 + x_2) + (y_2 - y_1)(y_1 + y_2)}{2\rho} \quad \text{(III.6)}$$

La représentation «$G\theta\rho$» (centre de gravité, orientation, longueur) sépare bien les paramètres indépendants: si l'on suppose que la variance des paramètres de la droite support est isotrope (c'est-à-dire que l'erreur est la même sur un point parallèlement et perpendiculairement à la droite support), la matrice de covariance associée au vecteur $(x_G, y_G, \theta, \rho)^T$ est diagonale [Deriche 90], donc les quatre paramètres sont indépendants. C'est celle que nous retenons.

Modèle dynamique d'un segment image

Deux approches sont possibles pour la modélisation d'un segment image en cours de suivi:

- Utilisation de la position et du mouvement du segment 3D correspondant. L'évolution du segment image est alors recalculée par projection dans le plan image. Cette approche est notamment utilisée par Frau et Llario [Frau 90].
- Utilisation de l'information 2D de l'image uniquement. Le modèle du mouvement est bidimensionnel: seul le mouvement apparent de la projection du segment dans l'image est modélisé.

Dans les deux cas le mouvement du segment est en général linéarisé au second ou au troisième ordre (vitesse ou accélération localement constante).

Le modèle que nous utilisons est très simple: nous modélisons le mouvement plan du segment image en considérant sa dérivée au premier ordre localement constante. Le modèle est donc composé à chaque instant du segment et de sa dérivée.

Le mouvement d'un segment $S = (X_g, Y_g, \theta, \rho)^T$ est supposé à vitesse localement constante. Il est donc représenté à chaque instant par le vecteur vitesse $\dot{S} = (\dot{X}_g, \dot{Y}_g, \dot{\theta}, \dot{\rho})^T$.

Au début d'une séquence de suivi, $\dot{S}$ est supposé nul; il est ensuite mis à jour en mesurant le déplacement effectif d'un segment après sa mise en correspondance.

III.2.2 Domaine de recherche

Connaissant le modèle $(S, \dot{S})$ d'un segment à un instant donné, on peut calculer la position prédite de ce segment à l'instant suivant par un développement au premier ordre (On considère que $\delta t = 1$):

$$\hat{S} = S + \dot{S}$$

Il n'est pas nécessaire de prédire la valeur de $\dot{S}$.

Les segments candidats seront recherchés à l'intérieur d'un domaine situé «autour» de cette prédiction. La définition de cette zone dépend du modèle d'incertitude utilisé.

Nous utilisons un modèle relativement simple: nous considérons que l'incertitude sur chaque paramètre est constante et que les quatre paramètres sont indépendants.

Un segment de la nouvelle image se trouve à l'intérieur du domaine de recherche si aucune de ses coordonnées ne s'écarte de la prédiction de plus de l'écart maximal admis.

Malgré la simplicité de ce modèle, il est optimal dans les limites du modèle linéaire retenu. En effet ce modèle est équivalent à celui obtenu en effectuant le calcul d'une zone de confiance à l'aide d'un filtre de Kalman dont l'état serait le modèle courant et les mesures successives les positions observées du segment.

Dans ce cas, Crowley utilise le fait que les variances sur la prédiction et sur l'état, ainsi que le gain du filtre, ne dépendent pas des observations [Crowley 88]. Par conséquent ceux-ci sont constants pour un segment donné. La variance sur la prédiction définissant la zone de recherche, celle-ci est également constante.

La taille de la zone de recherche s'exprime plus naturellement dans le domaine (x, y, θ, ρ) que dans le domaine des variances sur l'état ou la mesure; c'est pourquoi nous avons choisi cette approche.

III.2.3 Extraction des segments correspondants

Grâce à un *h-coding*[1] des segments sur les quatre paramètres du modèle «$G\theta\rho$» réalisé lors de leur extraction, l'accès aux segments proches du domaine de recherche est immédiat. Un balayage linéaire des segments extraits des bacquets du h-coding permet ensuite de vérifier l'appartenance de chaque segment au domaine et calculer le score d'appariement.

Le score d'appariement est ici la distance euclidienne dans l'espace (x, y, θ, ρ) qui sépare le segment de sa prédiction. Cette distance peut être définie comme la distance de Mahalanobis associée au modèle d'appariement probabiliste équivalent ; elle permet donc de réaliser des appariements optimaux au sens probabiliste.

Il est possible de réaliser des correspondances uniques en choisissant comme unique segment correspondant celui qui réalise le meilleur score. Il nous semble néanmoins préférable d'incorporer d'autres critères issus des autres contraintes du problème initial dans la phase de sélection des hypothèses : stéréovision, appartenance à un objet commun, critères photométriques, compatibilité entre plusieurs appariements...

C'est pourquoi le module de suivi présenté ici peut, selon le choix de l'opérateur, fournir une liste de candidats à l'appariement définie soit par un seuil sur le nombre maximal de candidats, soit par un seuil sur le score. D'autres solutions sont envisageables, notamment la sélection des appariements qui ont un score nettement détaché par rapport à la moyenne.

III.2.4 Résultats

Les résultats présentés ici sont obtenus en utilisant le module de suivi seul. Deux cas sont présentés :

1. une mire où les segments sont suffisament distincts pour éviter tout risque de confusion dans les appariements : le mouvement est un mouvement de translation de la caméra perpendiculairement à la mire. Entre la première et la dernière image de la séquence, la caméra a parcouru 2.50 m et 30 images ont été prises.

[1] Voir glossaire.

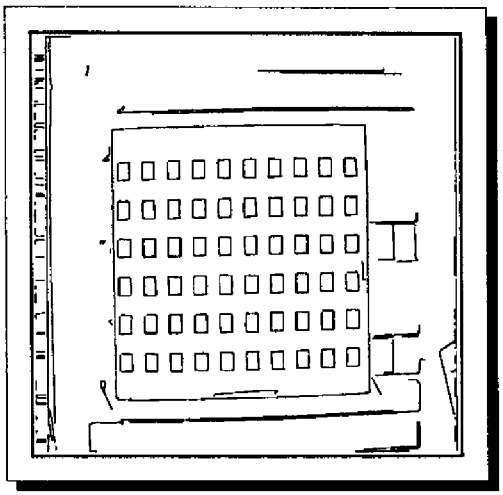


Figure III.3: *Image initiale de la mire*

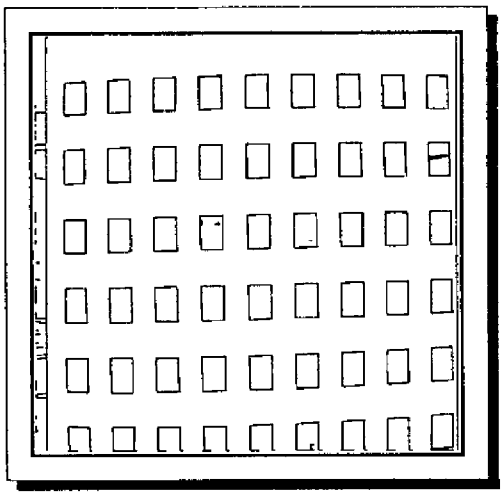


Figure III.4: *Image finale de la mire*

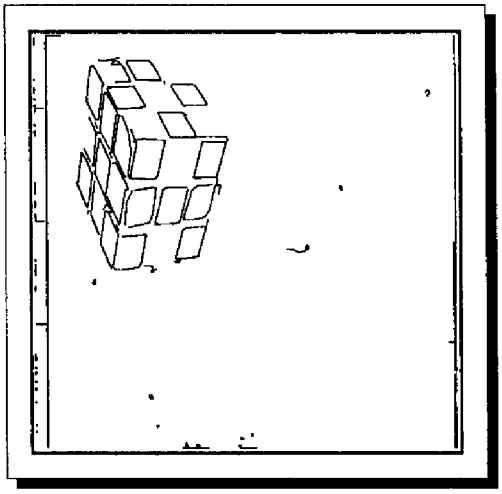


Figure III.5: *Image initiale du cube*

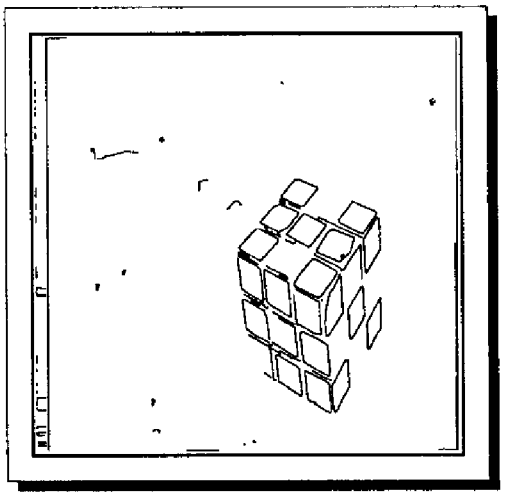


Figure III.6: *Image finale du cube*

2. une image réelle d'un Rubik's cube où un certain nombre d'ambiguïtés sont présentes : le suivi seul fournit dans un certain nombre de cas plusieurs appariements potentiels parmi lesquels se trouve le bon, mais où il n'est pas forcément le meilleur au sens du critère d'appariement défini. Nous verrons par la suite quelles autres techniques peuvent être utilisées pour discriminer entre ces appariements potentiels.

   Ici le mouvement était la combinaison d'une translation parallèlement au plan image combinée à une rotation du cube sur lui-même. La translation du cube

Image	Nombre de segments de l'image initiale	Pourcentage moyen de segments suivis
Mire	356	0.83
Cube	143	0.68

Figure III.7: *Résultats du suivi*

était d'environ 1 à 2 cm entre chaque image et la rotation (autour d'un axe à peu près parallèle à l'axe optique) avait une ampleur d'une dizaine de degrés (le cube a fait un tour complet sur lui-même). Trente image ont été prises.

Le tableau de la figure III.7 montre dans chaque cas le pourcentage moyen de segments suivis d'une image à l'autre: la moyenne des rapports entre le nombre de segments extraits de chaque image et le nombre de segments ayant trouvé un correspondant dans celle-ci. Ce taux est très élevé dans le cas de la mire en raison du faible nombre de segments parasites.

Dans le cas du cube, les mauvais appariements font baisser le taux moyen de segments suivis.

III.2.5 Discussion

Il y a deux causes aux problèmes rencontrés dans le cas de faux appariements ou de correspondances qui ne sont pas trouvées.

1. L'instabilité de la description de la scène construite à partir des segments de droite. On observe en effet dans les séquences d'images ou bien dans les triplets stéréo des variations apparemment aléatoires des segments: cassures, variations de longueur, disparitions....

   Ces variations sont liées à la complexité du processus d'extraction des segments. Les nombreuses heuristiques et approximations qui sont utilisées n'ex-

pliquent pas l'instabilité du processus mais rendent impossible une modélisation précise.

L'utilisation d'opérateurs de gradients plus sophistiqués, tels que celui de Deriche ou celui de Shen améliorent légèrement la qualité des segments obtenus, mais montrent que ce n'est pas le calcul du gradient qui est déterminant dans l'ensemble du processus.

2. Les erreurs dues à la simplicité du modèle cinématique utilisé: le mouvement des segments image peut s'écarter beaucoup du mouvement à vitesse localement constante par suite de variations du mouvement 3D, ou même de configurations particulières de celui-ci. Il y a des mouvements (la rotation sur elle-même de la caméra par exemple) qui produisent de grands mouvements dans l'image pour une amplitude faible du mouvement réel du robot.

   La modélisation 2D du mouvement des segments nous semble toutefois suffisante dans une grande majorité de cas, en raison du gain important en complexité par rapport au maintien d'un modèle de mouvement tridimensionnel. Néanmoins les techniques de reprise d'erreur introduites dans les deux applications du suivi font appel à une modélisation 3D des segments pour lesquels le suivi a échoué.

Cependant, malgré ces difficultés, notre approche du suivi de segments est largement validée par l'ensemble des résultats obtenus: la proportion de segments suivis correctement est importante et les résultats 3D obtenus (voir § III.3.5 et § IV.4) sont bons.

III.3 Suivi d'objet 3D

Dans cette partie le système de suivi d'objet 3D réalisé à partir du module de suivi de segments image est présenté en détails. Ce système, intégré au robot Hilare 1.5, lui permet de se diriger vers cet objet (éventuellement en mouvement), tout en conservant ses capacités traditionnelles de détection et d'évitement d'obstacles à l'aide des capteurs ultra-sons [Herrb 89].

Nous considérons un robot mobile dont la tâche est de trouver (en utilisant la vision) et éventuellement de prendre un objet — dont la position n'est pas connue a priori — dans une pièce. Cette tâche peut se décomposer en :

- Reconnaître et localiser l'objet, on suppose alors que le robot en possède un modèle.
- Se diriger vers l'objet en bouclant sur son suivi par la vision pour calculer les déplacements du robot.
- Quand le robot est suffisamment proche, s'arrêter pour éventuellement engager une autre action (la préhension par exemple).

Si le robot rencontre des obstacles durant son déplacement, il les évite. Mais pour poursuivre sa tâche principale (se diriger vers un objet), il maintient la caméra pointée sur celui-ci.

Le système complet est présenté dans [Noreils 90b]. Nous nous intéressons ici plus en détail à la partie vision.

III.3.1 Introduction

La vision réalise trois fonctions dans ce système: la *reconnaissance* initiale de l'objet dans la scène, le *suivi* des segments de l'objet au cours du déplacement du robot et la *localisation* de l'objet dans le repère lié à la caméra à partir de son modèle.

Contrairement à d'autres [Tsuji 85], nous ne cherchons pas à calculer le mouvement 3-D du robot à partir des déplacements observés dans le plan image. Au contraire nous utilisons le modèle connu de l'objet suivi pour localiser la caméra, et

donc le robot — grâce au calibrage de la position de la caméra par rapport au robot (voir § V.4).

Ces trois fonctions utilisent les segments des images comme primitives, extraits comme montré au chapitre II.

Nous avons retenu un modèle d'objet simple afin que les temps de calculs nécessaires à la reconnaissance et à la localisation restent raisonnables dans le cadre d'une démonstration avec le robot Hilare.

III.3.2 Reconnaissance de l'objet

La méthode de reconnaissance d'objet que nous utilisons ne prétend pas à l'universalité mais plutôt à la rapidité et à la simplicité.

Nous utilisons un objet plan, modélisé par un ensemble de chaînes fermées de segments de droite orientés (polygones). Ce modèle contient à la fois des informations nécessaires pour la reconnaissance et la localisation (Modèle de type CAO, arêtes/sommets).

Le module de reconnaissance s'appuie sur les trois éléments suivants du modèle :

- nombre de segments de chaque polygone,
- sens de rotation aux sommets des polygones,
- relations géométriques ou topologiques simples entre polygones.

Le module de reconnaissance utilise par ailleurs la localisation pour valider les hypothèses et lever les ambiguïtés dans les appariements.

Par exemple, l'hexagone présenté figure III.8 est défini par :

- deux chaînes polygonales de six segments,
- la seconde est incluse dans la première,
- orientées selon deux sens opposés (par rapport au gradient).

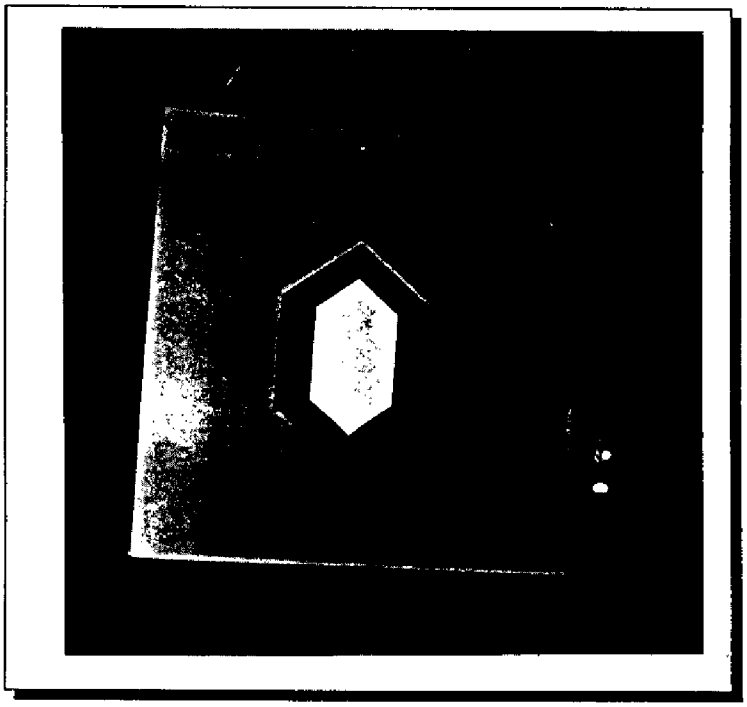


Figure III.8: *Objet utilisé pour le suivi*

Toutes ces caractéristiques sont topologiques et donc indépendantes du point de vue (invariants).

La procédure de reconnaissance de l'objet se décompose en trois phases, après extraction des segments de contour :

- Détection des chaînes fermées de segments [Pampagnin 90].
- Sélection des ensembles de polygones satisfaisant les conditions du modèle :
  - composées du bon nombre de segments
  - vérifiant les contraintes topologiques.
- Enfin, vérification et choix entre les hypothèses possibles par localisation. En effet, il reste en général plusieurs mises en correspondance possibles entre le modèle et la perception qui vérifient les propriétés topologiques du modèle. En utilisant le critère de qualité defini au paragraphe III.3.4, la correspondance

aboutissant à la meilleure localisation est conservée. Si aucune localisation n'atteint un seuil de qualité minimal, la reconnaissance échoue.

En phase initiale, s'il existe dans l'image plusieurs ensembles de polygones vérifiant toutes ces conditions, le robot demande à l'opérateur quelle est l'instance de l'objet qu'il doit suivre parmi celles qu'il a trouvé.

Cette méthode supporte bien les petites erreurs de perception (extrémités des segments non présentes ou segments cassés) grâce à une procédure de fusion des segments colinéaires et de « réparation » des jonctions, mais tous les segments de l'objet doivent être vus. Elle est donc robuste pour des objets bien contrastés comme celui que nous avons choisi. Il est envisageable de remplacer cet algorithme par un système de reconnaissance beaucoup plus général (objet polyèdrique, partiellement observé) [Pampagnin 90].

Une fois l'objet reconnu, l'ensemble des segments image qui le composent est mémorisé dans une structure particulière appelée par la suite *modèle image de l'objet.*

III.3.3 Suivi de l'objet

Le module de suivi permet de mettre à jour le modèle image de l'objet pour pouvoir le localiser sans avoir à lancer le processus de reconnaissance sur chaque image.

Le suivi est réalisé en recherchant dans chaque nouvelle image des correspondants aux segments du modèle image de l'objet extrait de l'image précédente.

Pour cela le formalisme de suivi de segments présenté au paragraphe III.2, sert de base ; on procède en trois étapes :

- Extraction des segments dans la nouvelle image à l'intérieur d'une fenêtre couvrant les positions prédites des segments du modèle image de l'objet.
- Recherche des appariements potentiels pour chaque segment du modèle image de l'objet.
- Validation des appariements potentiels par recherche de cliques maximales dans le graphe de compatibilité défini ci-dessous.

Deux appariements sont compatibles —compte tenu de la rigidité de l'objet et en supposant des petits déplacements d'une image à l'autre— si le déplacement relatif du segment modèle au segment image est le même pour les deux appariements.

Formellement, les appariements (S_{1i}, S_{2j}) et (S_{1k}, S_{2l}) sont compatibles si :

$$\theta_{1i} - \theta_{2j} = \theta_{1k} - \theta_{2l}$$

et :

$$|G_{1i}G_{2j}| = |G_{1k}G_{2l}|$$

Dans le graphe de compatibilité, la plus grande clique maximale (ensemble de sommets tous connectés) est ensuite cherchée en utilisant l'algorithme de Bolles [Bolles 82] : elle correspond au plus grand ensemble d'appariements tous mutuellement compatibles.

```
MaxClique(clique, candidats, cand_u_susp)
    x ← premier_element(cand_u_susp)
    pour chaque sommet y de candidats ∩ voisins(x) faire :
        clique ← clique ∪ {y}
        candidats ← (candidats ∩ voisins({y})) − {y}
        cand_u_susp ← cand_u_susp ∩ voisins({y})
        si cand_u_susp = ∅ et clique ≠ ∅ alors :
            clique est une clique maximale
        sinon si candidats ≠ ∅
            MaxClique(clique, candidats, cand_u_susp)
        finsi
    fin pour
fin
```

Figure III.9: *L'algorithme de calcul des cliques maximales ; on l'appelle initialement par :* `MaxClique(`$\emptyset$, *Ensemble des noeuds, Ensemble des noeuds)*

Si le nombre de sommets de la clique maximale est trop faible ou que le score moyen des appariements qui la composent est trop faible, le suivi échoue.

En pratique, grâce à l'utilisation de la fenêtre de recherche qui limite le nombre d'appariements potentiels, on n'a en général qu'une seule clique maximale dans le graphe de compatibilité. La recherche est alors linéaire par rapport au nombre d'appariements.

Cette méthode permet de discriminer, par ajout d'une contrainte globale issue d'une connaissance sémantique du modèle, entre les appariements potentiels ambigus. La seule hypothèse que l'on fait est que les segments suivis appartiennent à un objet rigide; la connaissance du modèle géométrique (CAO) de l'objet n'est pas utilisée ici.

Si l'objet est partiellement observé, cette méthode permet de suivre correctement les segments encore visibles.

Perte de segments et faux appariements

Si un segment du modèle image de l'objet ne trouve pas de correspondant dans la nouvelle image, une marque particulière *(nil)* est mise à sa place dans le modèle image de l'objet produit. Après la localisation, le segment 3D correspondant du modèle 3D sera reprojeté dans l'image et intégré au modèle image de l'objet.

Par ailleurs, lors de la localisation, d'éventuels faux appariements peuvent également être détectés et corrigés.

III.3.4 Localisation de l'objet

La méthode de localisation 3D de l'objet s'appuie sur les sommets de l'objet. Les segments du modèle image sont en correspondance avec les segments du modèle de l'objet grâce à la reconnaissance et au suivi. Les sommets sont calculés par intersection des segments des polygones du modèle image et sont donc directement mis en correspondance avec ceux du modèle de l'objet.

Nous présentons brièvement ici cette méthode sous-optimale mais rapide, inspirée de [Tsai 87].

Notations et équations

Soient $p_k = (i_k, j_k), k = 1, ..., n$ les sommets image et $m_k = (x_k, y_k), k = 1, ..., n$ les sommets correspondants du modèle (l'objet est plan). Nous utilisons un modèle «sténopé[2]» (projection directe) de la caméra qui est modélisée par sa matrice de projection (paramètres intrinsèques) [Faugeras 87b].

[2] Voir glossaire.

$$\Pi = \begin{pmatrix} \alpha_i & 0 & i_0 & 0 \\ 0 & \alpha_j & j_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

(voir figure IV.2, page 67)

Nous cherchons à trouver la transformation Tr (composée d'une rotation R suivie d'une translation T) qui vérifie au mieux les équations suivantes (exprimées en coordonnées projectives[3]) :

$$\begin{pmatrix} \alpha.i_k \\ \alpha.j_k \\ \alpha \end{pmatrix} = \Pi \times Tr \times \begin{pmatrix} x_k \\ y_k \\ 0 \end{pmatrix}$$

qui expriment que les sommets p_k (dans un repère lié au plan de l'objet, tel que $z_k = 0$) correspondent aux points m_k transformés par Tr et projetés sur le plan image de la caméra.

En notant

$$R = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} \qquad T = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

$$\begin{pmatrix} X_k \\ Y_k \\ Z_k \end{pmatrix} = R \times \begin{pmatrix} x_k \\ y_k \\ 0 \end{pmatrix} + T = \begin{pmatrix} r_1.x_k + r_2.y_k + t_x \\ r_4.x_k + r_5.y_k + t_y \\ r_7.x_k + r_8.y_k + t_z \end{pmatrix}$$

On obtient les équations

$$\begin{cases} \alpha_i.X_k &= (i_k - i_0).Z_k \\ \alpha_j.Y_k &= (j_k - j_0).Z_k \end{cases} \tag{III.7}$$

d'où l'on tire

$$\alpha_i.X_k.(j_k - j_0) - \alpha_j.Y_k.(i_k - i_0) = 0 \tag{III.8}$$

qui peut s'écrire comme une équation linéaire non homogène des inconnues $(\frac{r_1}{t_y}, \frac{r_2}{t_y}, \frac{r_4}{t_y}, \frac{r_5}{t_y}, \frac{t_x}{t_y})$. Ces cinq inconnues sont indépendantes malgré la contrainte d'orthogonalité de la matrice R.

[3]voir glossaire, *projection perspective*.

Algorithme

Il se décompose en trois étapes :

1. On résoud le système de n équations (III.8) par une méthode des moindres carrés. Notons que nous n'utilisont qu'une des deux équations fournies par chaque point, ce qui nécessite au moins cinq correspondances.

2. Les contraintes d'orthogonalité de R permettent de déduire de $(\frac{r_1}{t_y}, \frac{r_2}{t_y}, \frac{r_4}{t_y}, \frac{r_5}{t_y}, \frac{t_x}{t_y})$ deux valeurs possibles pour R, t_x et t_y (qui diffèrent par les signes de r_3, r_6, r_7, r_8).

3. Pour chacune de ces deux possibilités, la valeur de t_z est déterminée en utilisant les équations (III.7). On conserve celle qui donne le plus petit écart moyen sur les équations.

Le temps d'exécution de cette procédure est de moins de 0.1 seconde sur processeur *sparc*.

Mesure de la qualité de la localisation

On calcule après résolution l'écart moyen entre les sommets du modèle projetés sur l'image (compte tenu de la localisation) et ceux du modèle image de l'objet. Cet écart moyen est une bonne mesure de la qualité de la localisation, il est utile pour la reconnaissance et la détection de faux appariements.

Remise en état du modèle image

Une occultation ou une image perturbée peuvent conduire le processus de suivi à renvoyer un modèle image dégradé et faire échouer le suivi. L'utilisation de la connaissance du modèle de l'objet permet de détecter et de remédier à cette éventuelle dégradation.

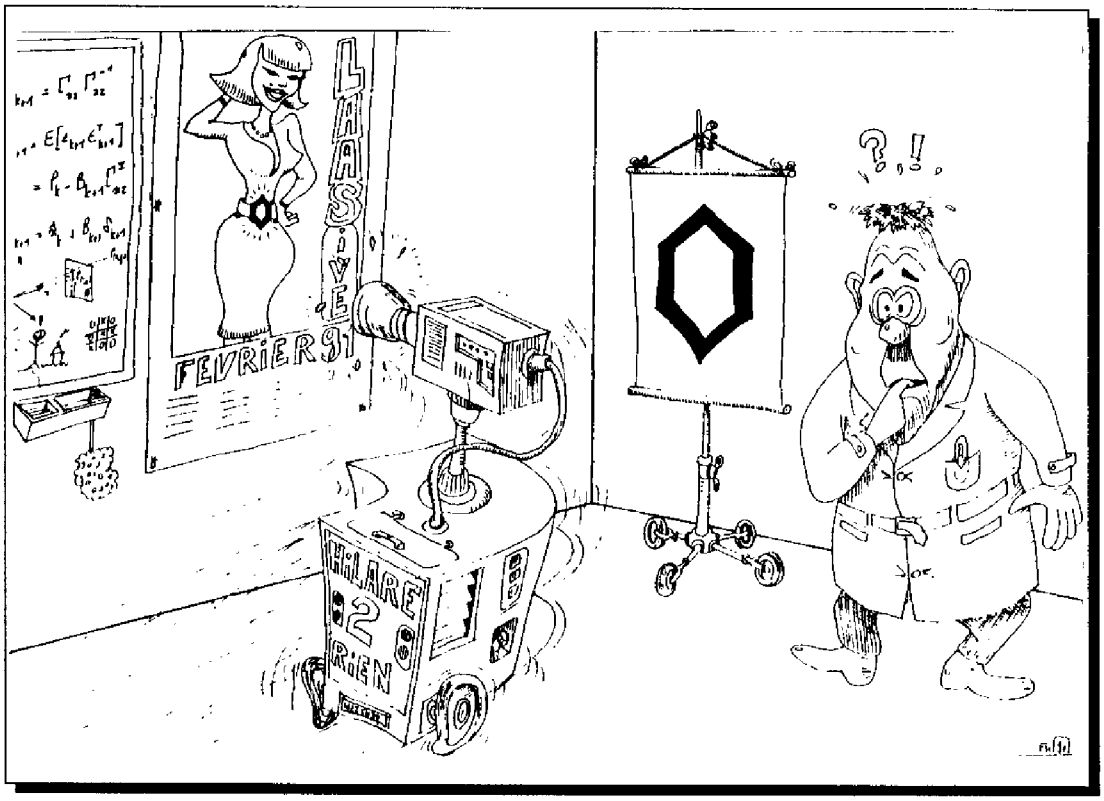
Faux appariements : Si la localisation basée sur le modèle image n'est pas assez bonne (qualité inférieure à un seuil), on cherche un faux appariement

éventuel. On tente des localisations en enlevant tour à tour un des segments, et on conserve le modèle image amputé qui a donné la meilleure localisation. Si aucune des nouvelles localisations n'atteint le seuil de qualité minimum, il doit y avoir plus d'un faux appariement et un constat d'échec est renvoyé.

Perte de segments : Lorsque des segments n'ont pas de correspondant (soit qu'il n'ait pas été trouvé lors du suivi, soit qu'un appariement défectueux ait été détecté et le segment enlevé), on effectue la localisation avec les sommets disponibles. S'il n'en reste pas assez — au moins cinq sont nécessaires —, un constat d'échec est renvoyé.

Une fois l'objet localisé, les segments manquants du modèle sont reprojetés dans le plan image pour compléter le nouveau modèle image.

On a ainsi au début de chaque itération un modèle image complet de l'objet qui garantit la robustesse du suivi.


LAAS
FEVRIER 91
HILARE
2
RIEN
?!

III.3.5 Résultats

Nous présentons de manière séparée les résultats du module de suivi d'objet proprement dit puis du module de localisation dans le repère caméra. Il faut cependant voir que les deux résultats sont liés : sans un suivi fiable, il est impossible d'avoir une localisation correcte (soit à cause d'un manque de données, soit à cause d'appariements incohérents), et réciproquement sans une localisation précise, il est impossible de maintenir un modèle toujours correct de l'objet par reprojection.

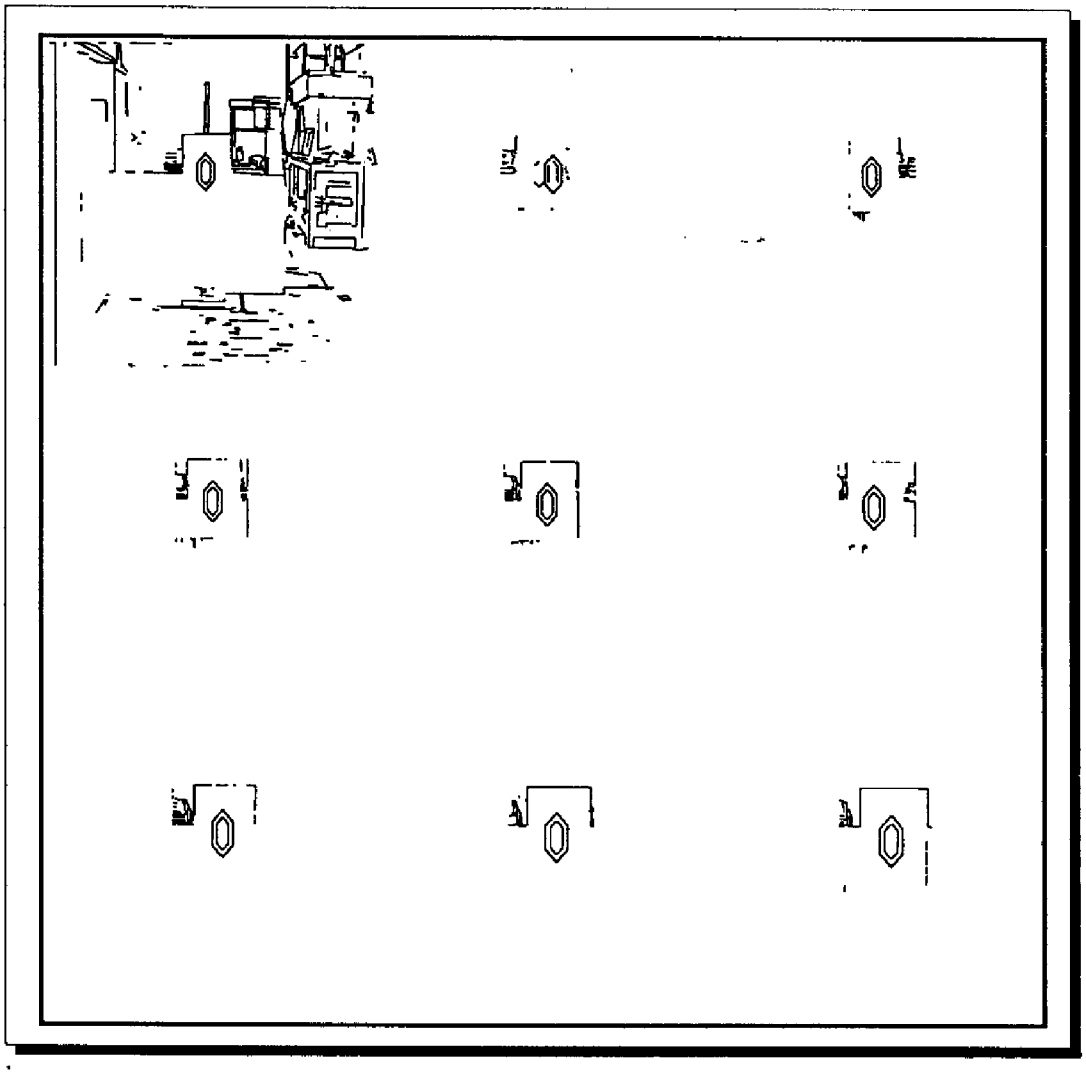


Figure III.10: *Une séquence de suivi de l'objet. En haut à gauche l'image initiale dans laquelle l'objet est reconnu, puis 8 images pour lesquelles on ne voit que la région d'intérêt.*

Suivi

Grâce à la méthode de reprojection des segments dans l'image et au bon contraste de l'objet utilisé, le suivi est très robuste : tant que l'objet se trouve suffisamment dans la zone de recherche du suivi, il est trouvé avec 100 % de réussite.

Il faut remarquer ici que le module de suivi seul - sans la sélection des appariements par recherche de clique maximale - échoue bien plus souvent, en raison de la grande ressemblance des segments sur les bords parallèles de l'objet. L'ajout de la recherche d'une clique maximale d'appariements compatibles est décisif pour la qualité des résultats.

Les seuls cas d'échecs sont ceux où le recentrage de la platine a échoué, conduisant l'objet à sortir entièrement de la zone de recherche.

Sur processeur *sparc*, le suivi et la localisation de l'objet prennent environ 0.7 secondes de temps CPU, sans optimisation particulière des calculs ou des données. Cela signifie que ce suivi peut être facilement utilisé dans un système temps réel à une cadence de plusieurs images par seconde en utilisant un système d'extraction des segments suffisamment rapide (le module d'extraction des segments sur architecture parallèle présenté au chapitre II n'est pas intégré ici).

Localisation

Le tableau de la figure III.11 résume les mesures effectuées sur la précision de la localisation. Les statistiques ont été réalisées sur 100 localisations successives de l'objet, placé à différentes distances et différentes orientations par rapport à la caméra.

On constate que l'erreur en x et en y (plan parallèle au plan image) est très faible (au maximum de quelques centimètres à six mètres). L'erreur en z augmente rapidement avec la distance (trente centimètres à six mètres), ce qui est normal.

L'orientation de l'objet influe beaucoup également sur la précision : les meilleurs résultats sont obtenus lorsqu'il est perpendiculaire à l'axe optique de la caméra.

	orientation de l'objet	
distance	90°	45°
2m	$\begin{pmatrix} 0.001 \\ 0.000 \\ 0.016 \end{pmatrix}$	$\begin{pmatrix} 0.002 \\ 0.001 \\ 0.017 \end{pmatrix}$
3m	$\begin{pmatrix} 0.002 \\ 0.000 \\ 0.019 \end{pmatrix}$	$\begin{pmatrix} 0.003 \\ 0.000 \\ 0.038 \end{pmatrix}$
5m	$\begin{pmatrix} 0.019 \\ 0.024 \\ 0.152 \end{pmatrix}$	$\begin{pmatrix} 0.028 \\ 0.011 \\ 0.283 \end{pmatrix}$
6m	$\begin{pmatrix} 0.030 \\ 0.000 \\ 0.286 \end{pmatrix}$	*échec reconnaissance*

Figure III.11: *Ecart-type (en mètres) sur la translation en fonction de la distance et de l'orientation de l'objet*

III.4 Conclusion

Dans ce chapitre nous avons présenté un module de suivi de segments de contour dans une séquence d'images. Ce module utilise une modélisation au premier ordre de l'évolution des segments en deux dimensions dans le plan image. Nous avons mis en évidence la validité du critère d'appariement choisi en le rapprochant du modèle de mise en correspondance probabiliste.

Dans la seconde partie de ce chapitre, nous avons montré une application de ce module de suivi au suivi d'objets dans l'espace tridimensionnel en l'intégrant à un système de reconnaissance d'un objet simple et un algorithme de localisation. La sélection entre les appariements potentiels est réalisée grâce à la contrainte de rigidité de l'objet suivi.

Les résultats présentés montrent le bon comportement de ce système tant en fiabilité du suivi qu'en précision de la localisation. L'intégration de ces fonctionnalités dans un robot mobile sera présentée au chapitre V.

A l'avenir ce système pourra être utilisé pour suivre des objets plus génériques (modèles réellement tridimensionnels, partiellement occultés...) en se basant sur les travaux de L.H. Pampagnin [Pampagnin 90] ou de P. Grandjean [Grandjean 90].

Chapitre IV

Stéréovision dynamique

La stéréovision passive est un moyen d'acquisition de données tridimensionnelles relativement intéressant et très utilisé en robotique. Ses principaux avantages sont la rapidité des acquisitions et son faible coût ; par ailleurs elle fournit des données significatives : par exemple les segments de contours tridimensionnels. Par contre, ses temps de calcul sont relativement longs, la précision en profondeur décroît avec le carré de la distance et son champ de vue est plutôt restreint.

Pour construire des cartes tridimensionnelles à la fois plus précises et couvrant un champ plus important, on peut fusionner plusieurs cartes stéréo obtenues séparement, en appariant les segments 3D communs à plusieurs vues. Cette technique reste néanmoins coûteuse en temps de calcul : il faut ajouter le temps de calcul des appariements entre segments tridimensionnels et celui de la fusion.

Crowley [Crowley 90] utilise le *token-tracker* développé au LIFIA dans le cadre du projet Esprit DMA (P 940) pour suivre les segments verticaux dans chaque image de la stéréo de manière indépendante et les fusionne avec le modèle 3D en deux temps : *1.* reconstruction d'un modèle 3D local, *2.* fusion dans le modèle global.

Nassab et Faugeras [Faugeras 89] utilisent le filtre de Kalman étendu pour fusionner directement les segments image 2D des différentes vues dans le modèle 3D.

en tenant compte du déplacement du robot.

Matthies utilise une stéréovision dense dans l'image (qui prend en compte tous les pixels) pour construire une carte de profondeur dense [Matthies 89].

L'utilisation de la vision dynamique permet, en conservant les appariements stéréo d'une vue à l'autre, d'éviter à la fois le calcul de nombreuses stéréos et le calcul des appariements en 3D. De plus, la fusion numérique des observations peut être faite directement entre le segment 3D et les différentes observations 2D, tout en prenant en compte d'autres mesures sur le déplacement du robot (odométrie) ou sur l'environnement (télémètre Laser...).

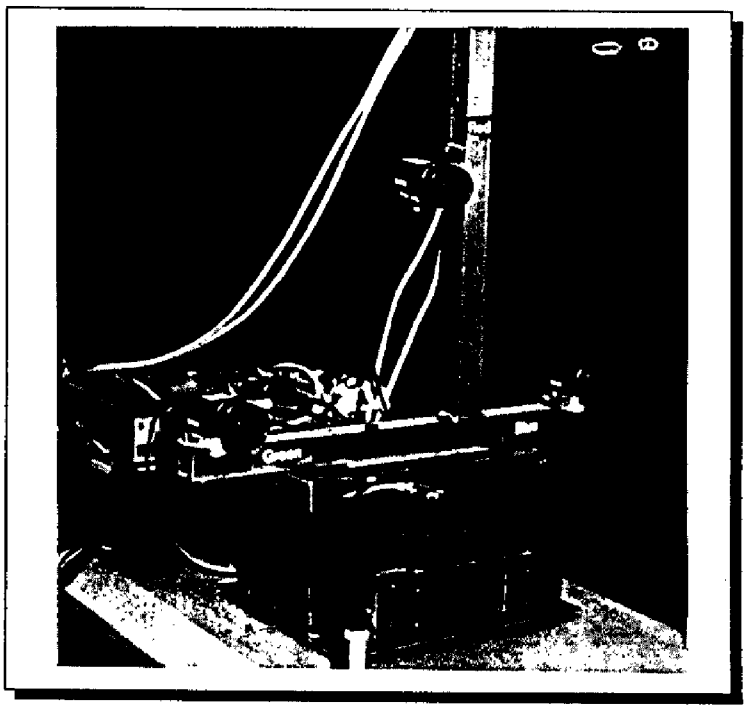


Figure IV.1: *Le banc de stéréovision d'Hilare 1.5. La distance entre les caméras est environ de 50cm*

IV.1 La stéréovision trinoculaire

Nous utilisons un algorithme de stéréovision trinoculaire développé à partir de l'algorithme binoculaire initialement mis au point par A. Robert de Saint Vincent [Robert 86].

IV.1.1 La géométrie du système

Le banc de stéréovision utilisé est composé de trois caméras placées en triangle conformément à la photo IV.1. Rappelons rapidement le modèle géométrique de ce système :

La formation de l'image à l'intérieur de chaque caméra est représentée par un modèle de projection perspective (appelé modèle du *sténopé* — figure IV.2), caractérisé par un centre optique O, un plan de projection $\mathcal{P}$ et une distance focale. Ces éléments composent les paramètres intrinsèques de chaque caméra.

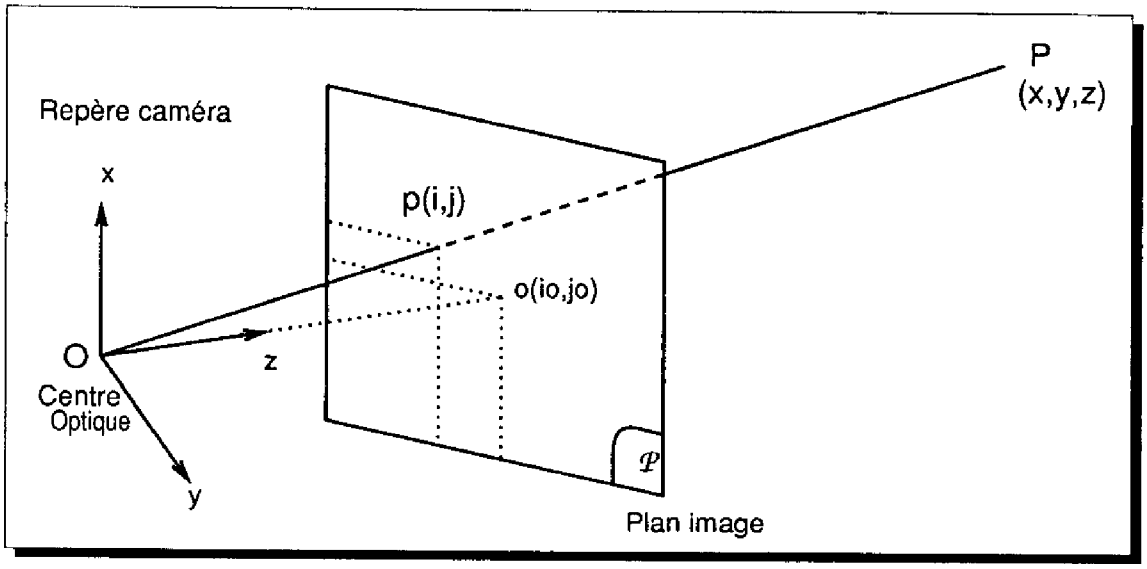


Figure IV.2: *Le modèle de projection utilisé pour une caméra*

Les paramètres extrinsèques définissent quant à eux la position de chaque caméra dans un repère commun.

L'ensemble de ces paramètres est estimé par le calibrage du système de stéréovision [Robert 86], réalisé hors-ligne.

Ce modèle physique est représenté par une matrice 3x4, appelée *matrice perspective* notée Π.

Un point de l'espace tri-dimensionnel $P = (x, y, z)^T$ est vu dans l'image au point $p = (i, j)^T$, intersection de la droite OP du plan $\mathcal{P}$. On a la relation suivante en coordonnées projectives[1] :

$$\begin{pmatrix} \alpha i \\ \alpha j \\ \alpha \end{pmatrix} = \Pi \times \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

La matrice Π se décompose en un produit de deux matrices 3x4, la matrice I des paramètres intrinsèques et la matrice E des paramètres extrinsèques :

$$\Pi = I \times E$$

La matrice I est de la forme

$$I = \begin{pmatrix} f_i & 0 & i_0 & 0 \\ 0 & f_j & j_0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

où f_i et f_j sont les «facteurs d'échelle» suivant les axes i et j de la caméra (traduction du fait que les pixels peuvent ne pas être carrés). Le pixel (i_0, j_0) est l'origine de l'image, c'est à dire la projection du centre optique de la caméra. La matrice I est la matrice de projection dans le repère lié à la caméra dont le centre optique est l'origine et dont l'axe de visée est l'axe z.

La matrice E est une matrice de transformation homogène qui permet de passer du repère dans lequel on souhaite travailler (commun aux trois caméras) à celui lié à la caméra.

Dans ce modèle, nous ne prenons pas en compte d'éventuelles distorsions de l'image qui se produisent notamment si l'on utilise des objectifs de courte focale. Il existe des modèles de calibrage qui traitent ce problème [Orteu 90]. Néanmoins dans ce modèle les lignes épipolaires ne sont plus des droites, ce qui pose de nouveaux problèmes.

[1] voir glossaire, *projection perspective*.

IV.1.2 Les vraies et les fausses contraintes

Le problème de la recherche des appariements entre les segments des images d'un triplet ou d'une paire stéréo est relativement difficile ; nous disposons d'un certain nombre de contraintes pour sélectionner les bons appariements parmi l'ensemble des triplets de segments.

Il faut toutefois remarquer que les contraintes habituellement utilisées ne correspondent que plus ou moins à la réalité de la formation des segments dans les images. Celles-ci peuvent être rangées en 4 catégories, que nous avons classées par ordre de réalité :

1. Contrainte épipolaire
2. Contrainte de continuité
3. Contrainte de ressemblance
4. Contrainte d'ordre

La contrainte épipolaire est une contrainte géométrique pure, et c'est la plus forte de toutes les contraintes. Elle exprime que les points appariés entre deux images sont les projections d'un même point de l'espace tridimensionnel. Appliquée aux extrémités des segments, elle permet de contraindre fortement la zone de recherche. Dans le cas trinoculaire, l'appariement de deux segments dans les deux premières images fournit une seule position possible pour la troisième projection. Il est donc très facile de valider un appariement partiel. La figure IV.3 illustre cette contrainte.

Dans le cas général où les axes optiques des caméras ne sont pas parallèles, les lignes épipolaires correspondant à deux points d'une image ne sont pas parallèles : elles appartiennent à un même faisceau de droites dont l'équation est donnée à partir des matrices de calibrage. Cependant, on peut se ramener au cas des lignes épipolaires parallèles par *rectification* des images à partir du calibrage du système stéréo [Ayache 89].

Shen, Castan et Zhao ont montré que l'utilisation de trois caméras et le choix d'ensembles de points commes primitives à apparier permet peut éviter efficacement

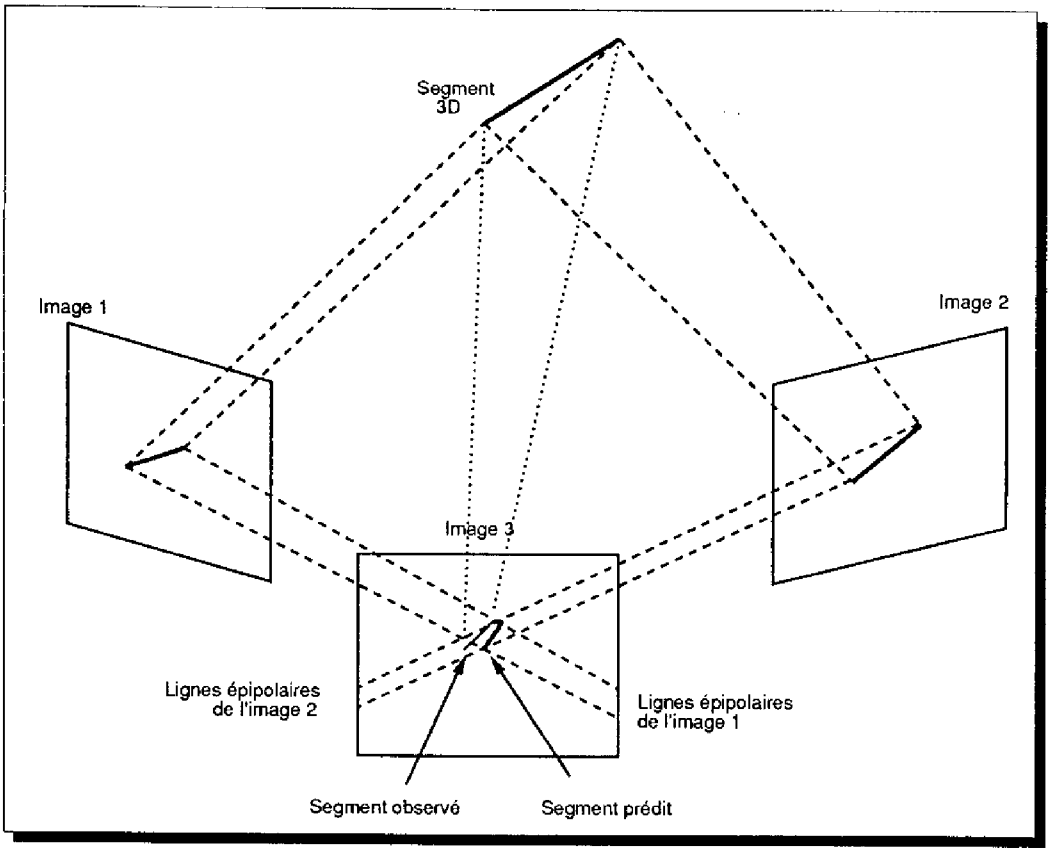


Figure IV.3: *La contrainte épipolaire en stéréovision trinoculaire : les segments trouvés dans les images 1 et 2 contraignent la position de la troisième projection. Si on trouve un appariement possible à cet endroit, le triplet peut être accepté.*

les ambiguités d'appariement sans utiliser aucune connaissance a priori ni aucune contrainte heuristique sur la scène [Shen 90].

La contrainte de continuité exprime que des segments voisins dans une image ont des disparités voisines. Ceci est évidemment généralement faux et peut conduire à de nombreuses erreurs. Néanmoins une version restreinte de cette contrainte est très souvent vraie : au voisinage d'un segment situé à une profondeur donnée, on en trouve d'autres avec une disparité comparable. Cette contrainte exprime que dans les environnements traités un segment se trouve rarement isolé sans voisins.

Les contraintes de ressemblance entre segments peuvent être de plusieurs natures : géométrique (longueur, orientation...) ou photométrique (luminance, colo-

rimétrie...). Ces contraintes sont fausses à plusieurs niveaux : au niveau géométrique, une arête 3D peut donner deux images fort dissemblables malgré un écart assez faible entre les caméras : lignes fuyantes, occultations partielles ; au niveau photométrique, les reflets de type spéculaire peuvent introduire de forts écarts. Enfin, l'instabilité de l'extraction des segments peut introduire des dissemblances liées par exemple au découpage de chaînes de contours courbes. Ce dernier défaut est surtout sensible en environnement extérieur qui est plus difficilement décrit par des segments de droites.

La ressemblance peut également être caractérisée en utilisant la notion d'organisation perceptuelle : des segments correspondants sont contraints à appartenir aux mêmes types d'organisations perceptuelles [Skordas 88] (par exemple, deux segments colinéaires appariés à deux segments colinéaires). Cette contrainte, de type relationnel, permet de calculer les appariements définitifs par recherche de clique maximale dans le graphe de compatibilité entre appariements potientiels, construit sur la base des différentes contraintes.

La contrainte d'ordre est une contrainte artificielle introduite pour la stéréo binoculaire : elle indique que les segments des différentes images intersectent les lignes épipolaires dans le même ordre. Cette condition — qui est vérifiée strictement pour un objet convexe — permet de réduire considérablement la complexité des algorithmes d'appariement par techniques syntaxiques [Crowley 90], au prix de la perte de tout appariement dans les régions où elle n'est pas vérifiée. (Par exemple, l'ordre dans lequel sont vu les différents pieds d'une table ou d'une chaise peut être différent pour deux caméras). L'introduction de la contrainte épipolaire à trois caméras — plus robuste — permet à notre avis de supprimer cette contrainte de l'algorithme.

L'algorithme que nous utilisons est du type *prédiction-vérification* (figure IV.4). Il utilise les trois premiers types de contraintes présentés.

IV.1.3 Prédiction

La phase de prédiction des hypothèses utilise la géométrie épipolaire et la ressemblance des segments pour créer une liste d'appariements possibles entre les segments de deux images.

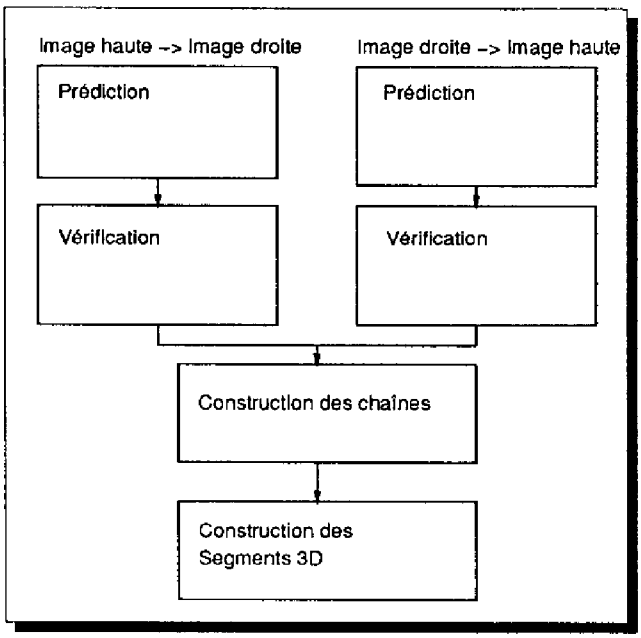


Figure IV.4: *Schéma de principe de l'algorithme de stéréovision*

Pour être candidat à l'appariement avec un segment S_1 de l'image 1, un segment doit obligatoirement couper le segment de ligne épipolaire correspondant au milieu de S_1 et limité par les disparités minimales et maximales admises.

La recherche des candidats est faite par un *h-coding* sur les segments issus des images, qui permet d'avoir très vite la liste des segments intersectant le segment de ligne épipolaire considéré.

Parmi cette liste, un tri peut être fait en utilisant une contrainte de ressemblance très lache, basée uniquement sur l'orientation et éventuellement la photométrie, avec des seuils de ressemblance très bas.

IV.1.4 Vérification

La phase de vérification utilise deux tests successifs pour valider les appariements réalisés lors de la prédiction :

Première vérification : contrainte épipolaire sur la troisième caméra

Le test le plus discriminant est celui de la contrainte épipolaire sur la troisième image : on recherche s'il existe dans la troisième image un segment passant par le

point d'intersection des lignes épipolaires associées aux milieux des deux segments appariés à valider.

Plus précisément le point de recherche est défini comme l'intersection de la ligne épipolaire associée au milieu de $S1$ dans l'image 3 d'une part, et de la ligne épipolaire correspondant à l'intersection de la ligne épipolaire associée au milieu de $S1$ dans l'image 2 et de la droite support de $S2$ d'autre part. (figure IV.3)

Deuxième vérification : contrainte de continuité de la disparité

La contrainte épipolaire testée ci-dessus pouvant se trouver réalisée de manière fortuite, un dernier test est fait sur la continuité relative de la disparité pour tous les appariements encore ambigus : parmi les voisins d'un segment on doit en trouver une proportion suffisante appariés avec une disparité voisine.

IV.1.5 Un sens puis l'autre

Pour garantir la symétrie des correspondances construites, l'algorithme procède en trois étapes successives :

1. recherche des correspondants des segments de l'image haute dans l'image droite en utilisant l'image gauche pour les vérifications.
2. recherche des correspondants des segments de l'image droite dans l'image haute en utilisant l'image gauche pour les vérifications.
3. fusion des deux correspondances partielles : durant cette phase, on vérifie la cohérence entre les appariements partiels, et on construit les appariements définitifs.

Cette méthode permet de construire des appariements où un segment est en correspondance avec plusieurs, dans le cas où un segment est coupé dans une image et pas dans l'autre et garantit la symétrie (par rapport à la paire de caméras principale) des appariements [Robert 86].

IV.2 Le suivi des appariements

Le suivi des appariements réalisés par la stéréo utilise le module de suivi général présenté au chapitre III. Pour chaque segment 2D des trois images, le module de suivi fournit une liste d'appariements potentiels.

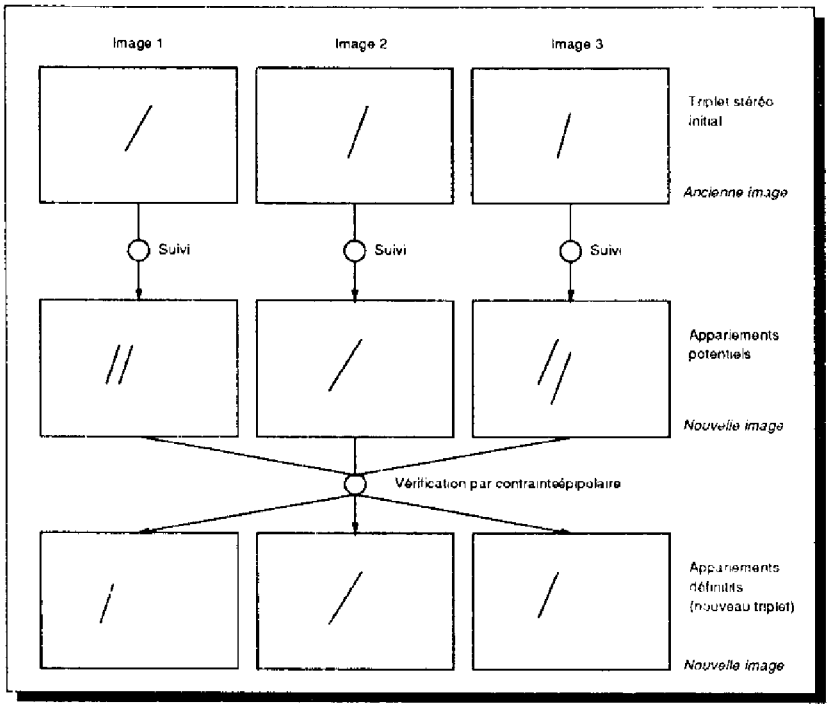


Figure IV.5: *Le suivi des appariements de la stéréo*

Ensuite, la contrainte de la géométrie épipolaire est utilisée pour ne garder que les segments appartenant à un triplet vérifiant la contrainte épipolaire (figure IV.5) : de même que lors de la vérification des appariements de la stéréovision, une paire de segments appariés entre les deux premières images définit un segment dans la troisième image. Si l'on trouve effectivement un segment proche du segment prédit dans la troisième image, l'appariement initial est validé ; dans le cas contraire, il est rejeté.

Une vérification supplémentaire peut être faite en considérant la continuité de la disparité pour un triplet d'une vue à l'autre : sachant que le mouvement du robot a été de faible amplitude entre les deux vues, la disparité (liée directement à la position du segment 3D associé) des appariements mesurée entre deux images n'a pas

pu changer beaucoup. On peut donc rejeter tous les triplets pour lesquels la disparité s'éloigne trop de la valeur précédente. L'absence de modèle 3D du mouvement au moment du suivi, interdit toutefois la prédiction de la nouvelle disparité.

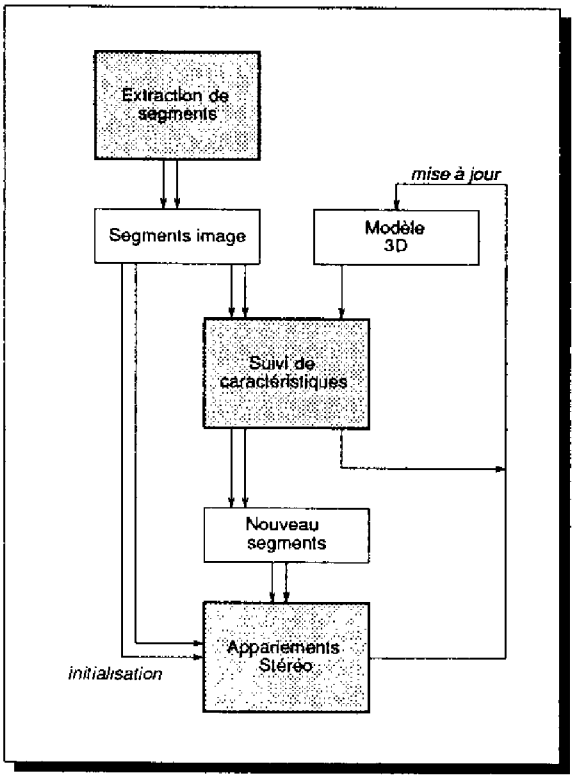


Figure IV.6: *Le système de stéréovision dynamique*

IV.3 Fusion des segments 2D et 3D

La stéréovision initiale fournit un ensemble de segments 3D. Le suivi permet d'obtenir une suite d'appariements stéréo dans la séquence d'images, sans avoir recours à l'algorithmique lourde de calcul des appariements sans information a priori.

Pour enrichir les données tridimensionnelles à partir de cette information nouvelle, nous avons choisi de fusionner directement les observations 2D successives avec le segment 3D de départ. Pour cela nous écrivons que le segment 3D se projette dans la nouvelle image (après transformation par le mouvement du robot) à la nouvelle position observée.

Cette relation nous fournit en fait l'équation de mesure d'un filtre dont l'état est le modèle 3D et les positions sucessives du robot, et dont les mesures sont les vues successives des segments en 2D.

Pour réaliser le filtrage, nous utilisons l'outil de filtrage général développé au LAAS.

La figure IV.6 présente cette méthode qui permet d'enrichir incrémentalement la précision de la localisation 3D des segments. Lorsque le nombre de segments du modèle suivis devient trop faible, la stéréovision est relancée sur les nouveaux segments 2D apparus dans les images. Ceux-ci créent de nouveaux éléments dans le modèle qui vont ensuite pouvoir être suivis.

IV.3.1 Le filtre généralisé pour la manipulation de données imprécises

La technique repose sur une représentation probabiliste du premier ordre (matrices de covariances) de la précision des vecteurs de mesure et des paramètres à estimer. Le filtre de Kalman permet, étant donné un vecteur d'état x et de variance V_x et un vecteur de mesure z de variance V_z indépendants liés par une relation linéaire $z = H.x$, de calculer une nouvelle estimée de x, ainsi qu'une nouvelle covariance intégrant l'information apportée.

Le filtre de Kalman peut être *étendu* à des relations implicites non linéaires $F(x, z) = 0$ en linéarisant autour de l'estimée courante. Il peut aussi être *généralisé* pour prendre en compte des bruits corrélés et colorés [Jazwinski 70].

Afin d'appliquer ces techniques au problème de la modélisation incrémentale de l'environnement par un robot mobile, P. Moutarlier, R. Chatila et P. Grandjean ont réalisé un développement permettant d'utiliser efficacement le filtrage de Kalman dans des cas non linéaires avec des bruits corrélés et colorés [Moutarlier 89b, Moutarlier 89c]. Ce filtre *généralisé* permet de prendre en compte explicitement les corrélations entre les différents vecteurs associés dans une équation générale de mesure :

$$F(x_1, x_2, \ldots, x_n) = 0$$

La consistance de l'ensemble des vecteurs est garantie par le maintien des matrices

de corrélation. En particulier il est alors possible de réestimer un vecteur non mesuré quand il possède une corrélation avec l'équation de mesure. Il est aussi possible de tester la correspondance de vecteurs pour une équation F en tenant compte des covariances et des corrélations.

Ce filtre est un estimateur linéaire optimal de l'état d'un système, à partir d'une mesure, même indirecte, de celui-ci : il a été établi, en reprenant la démarche utilisée par Kalman pour la minimisation du critère de variance de l'erreur sur un objet. Si l'on possède une estimation $\hat{x}_k$ optimale à l'instant k d'un état x, avec une erreur d'estimation $\epsilon_k = x - \hat{x}_k$, et qu'une nouvelle information est donnée à $k+1$ par une mesure z_{k+1} dont nous pouvons connaître une prédiction non biaisée $\hat{z}_{k+1}$, alors les équations suivantes permettent de calculer une nouvelle estimation de l'état :

$$\hat{x}_{k+1} = \hat{x}_k + B_{k+1}\delta_{k+1} \tag{IV.1}$$

$\delta_{k+1} = z_{k+1} - \hat{z}_{k+1}$ étant l'erreur de prédiction, Γ_{zz} sa variance et Γ_{xz} la covariance entre l'erreur de prédiction δ_{k+1} et l'erreur courante d'estimation ϵ_k. Le gain et la variance de la nouvelle erreur d'estimation sont donnés par :

$$B_{k+1} = \Gamma_{xz}\Gamma_{zz}^{-1}$$

$$P_{k+1} = E[\epsilon_{k+1}\epsilon_{k+1}^T] = P_k - B_{k+1}\Gamma_{xz}^T$$

Il est nécessaire pour mener ces calculs de disposer explicitement de toutes les corrélations C_{nm} entre les vecteurs d'erreur sur les composantes n et m de l'état ; ces corrélations doivent être mises à jour en même temps que l'état. Néanmoins, nous disposons également d'une version «réduite» du filtre qui permet de négliger ces corrélations, ce qui réduit considérablement la complexité des opérations de fusion.

IV.3.2 Modélisation de la stéréovision dynamique

En vue de l'utilisation du filtre généralisé pour fusionner les observations 2D des segments 3D issus de la stéréovision, nous utilisons la modélisation suivante du processus de suivi des segments :

Représentation des droites

Nous utilisons les représentations proposées dans [Ayache 88], qui sont bien adaptées à l'utilisation dans le filtre de Kalman, à savoir :

Droites 2D : Les droites du plan sont représentées dans deux *cartes*, à l'aide de deux paramètres (α, β) :

carte 0 : droites non parallèles à l'axe des y :

$$y = \alpha x + \beta$$

carte 1 : droites non parallèles à l'axe des x :

$$x = \alpha y + \beta$$

Droites 3D : Les droites dans l'espace sont représentées de la même manière dans trois cartes, à l'aide de quatre paramètres (a, b, p, q)

carte 0 : droites non orthogonales à l'axe des x :

$$\begin{cases} y = ax + p \\ z = bx + q \end{cases}$$

carte 1 : droites non orthogonales à l'axe des y :

$$\begin{cases} x = ay + p \\ z = by + q \end{cases}$$

carte 2 : droites non orthogonales à l'axe des z :

$$\begin{cases} x = az + p \\ y = bz + q \end{cases}$$

Dans les deux cas, les matrices de covariances associées aux vecteurs de paramètres sont incluses dans la représentation avec un indice indiquant le type de carte utilisé.

Ces représentations sont linéaires, non-ambiguës, complètes et dérivables.

Projection d'un segment 3D

Soient l_1, l_2, l_3 les trois lignes de la matrice de projection Π d'une caméra. Un point M de l'espace se projette en (i, j) dans l'image par les équations :

$$i = \frac{l_1.M}{l_3.M} \tag{IV.2}$$

$$j = \frac{l_2.M}{l_3.M} \tag{IV.3}$$

Une droite 3D est représentée par un point P et un vecteur V :

$$M = P + kV$$

En remplaçant M dans les équations IV.2 et IV.3, puis en éliminant k entre ces deux dernières, on obtient l'équation de la droite 2D correspondant à la projection de la droite 3D :

$$l_1.Vl_2.P - l_1.Pl_2.V + j(l_1.Pl_3.V - l_1.Vl_3.P) + i(l_2.Vl_3.P - l_2.Pl_3.V) = 0 \tag{IV.4}$$

Les coordonnées des extrémités du segment 2D sont obtenues à partir de la projection des extrémités du segment 3D. Seules les incertitudes sur les paramètres des droites support sont prises en compte dans le filtrage.

Néanmoins, dans la suite, nous appellerons — par abus de langage — *segments* les droites supports de ces segments. Notamment lorsqu'il est question de la variance d'un segment, c'est en fait de la variance des paramètres de sa droite support qu'il s'agit.

Equation de mesure

Considérons la transformation Tr entre l'ancienne et la nouvelle position du robot. Connaissant S_3 un segment 3D à l'ancienne position du robot et S_2 une nouvelle observation de ce segment dans la caméra i (projection Π_i) à la position courante du robot, nous écrivons l'équation :

$$\Pi_i(Tr(S_3)) - S_2 = 0 \tag{IV.5}$$

Cette équation est l'équation de mesure du filtre de Kalman étendu qui est utilisé pour la fusion.

Le filtre généralisé nécessite la connaissance des matrices jacobiennes de l'équation de mesure par rapport aux différentes variables. La projection Π_i est considérée comme certaine (le calibrage effectué hors-ligne n'est pas remis en cause). Il reste donc les Jacobiennes de la fonction par rapport à Tr et S_3 qui sont calculées numériquement pour chaque point par une approximation au premier ordre à partir de l'expression IV.4.

IV.3.3 Mise en œuvre

Pour chaque nouveau triplet d'images, on réalise les opérations suivantes :

1. On calcule les listes d'appariements potentiels dans les trois images en utilisant le module de suivi.

2. Les triplets d'appariements potentiels sont vérifiés en utilisant les contraintes présentées au § IV.2 (épipolaire).

3. La nouvelle position du robot est calculée par le filtre à partir d'une estimation initiale donnée par l'odométrie et du calibrage robot-repère stéréo (voir chapitre V).

4. Dans une seconde passe du filtre, les observations 2D des nouveaux segments sont fusionnées avec les segments 3D du modèle, sur la base des correspondances établies par le suivi.

   La fusion est réalisée en deux passes du filtre à cause du biais introduit par la position du robot : tous les segments étant exprimés dans le repère global, une erreur sur la position du robot affecte tous les segments. Pour éviter qu'une imprécision sur la localisation des premiers segments ne fasse trop dévier le filtre, la position du robot est mise à jour dans une première passe sans modifier les données du modèle. Lors de la deuxième passe, le biais sur la nouvelle position du robot est minimisé et n'affecte pas les mises à jour des segments du modèle.

5. Enfin, tous les segments du modèle qui n'ont pas été mis à jour (après échec du tracking des triplets stéréo correspondants) sont reprojetés dans les trois images pour permettre au suivi de garder le maximum d'hypothèses.

Toutes ces étapes sont facilement parallélisables, sauf peut-être la seconde si l'on veut garantir de manière stricte la cohérence globale des appariements (correspondance bi-univoque entre les anciennes et les nouvelles observations). Toutefois cette cohérence n'est pas indispensable. En effet les segments mal appariés ne peuvent en général pas être suivis en raison des contraintes géométriques fortes imposées. Par conséquent, l'influence d'une telle erreur reste limitée à une ou deux images et peut donc être négligée.

L'algorithme d'appariement de la stéréovision est relancé sur les segments non suivis lorsque le nombre de segments suivis devient trop faible. Selon le type d'application et les performances désirées cette fréquence peut être ajustée. Par ailleurs, il est envisageable de mémoriser les segments non suivis durant quelques itérations pour enrichir les nouveaux appariements obtenus par la stéréovision à partir d'un suivi «à l'envers» dans les images précédentes.

IV.4 Résultats

Des expérimentations ont été réalisées à partir de scènes réelles. Le banc de stéréovision du robot Hilare 1.5 a été utilisé pour acquérir les images. La figure IV.7 montre la scène initiale vue par les trois caméras et la figure IV.8 montre les appariements initiaux de la stéréovision. Une dizaine d'itérations du suivi on été réalisées. Le tableau IV.12 montre l'évolution de la variance moyenne sur les paramètres des droites 3D au fur et à mesure de la fusion. Enfin, les figures IV.9 et IV.10 montrent les images de segments 3D obtenues, après la stéréovision initiale et après 7 itérations de suivi. Le robot a parcouru 50 cm durant ces 7 itérations.

Dans cette expérimentation, la version réduite du filtre a été utilisée. Sur un processeur *sparc*, et alors qu'aucune optimisation particulière n'est réalisée, le suivi des segments sur les trois images et la sélection des appariements nécessitent au total environ 0.7 secondes, soit entre 3 et 4 millisecondes par segment suivi. La

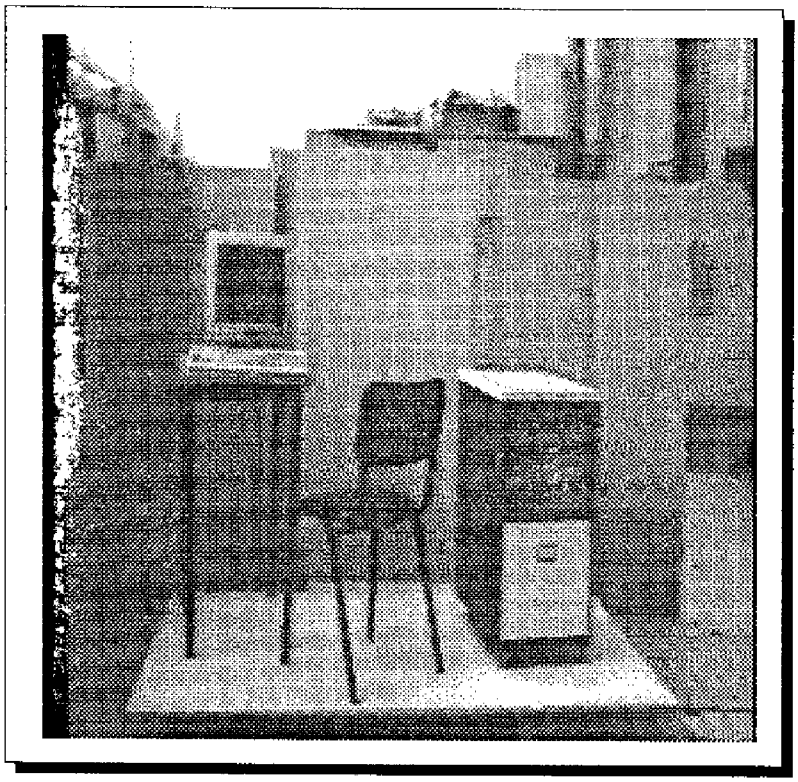


Figure IV.7: *La scène de bureau utilisée dans le suivi*

fusion nécessite quant à elle au total un peu moins de 4 secondes, ce qui représente environ 30 ms par segment, dans le cas d'une implémentation idéalement parallèle.

On constate — par une vérification manuelle — que le nombre de faux appariements est relativement faible. De plus, un faux appariement ne peut en général être suivi, car la vérification par la contrainte trinoculaire réussit rarement deux fois de suite « par hasard ». La présence de la reprojection des éléments perdus du modèle permet de récupérer de telles erreurs après la perte du suivi.

Par ailleurs, le nombre de segments suivis lors de chaque itération reste à peu près constant en mode permanent (voir figure IV.11). La forte baisse des deux premières itérations s'explique par la disparition du champ de vue d'une dizaine de segments d'intérêt.

La table de la figure IV.12 montre l'évolution de la variance des segments du modèle tridimensionnel en fonction du nombre de fois où ils ont été vus, c'est-à-dire du nombre d'observations fusionnées pour chacun d'eux. En donnant la valeur unité

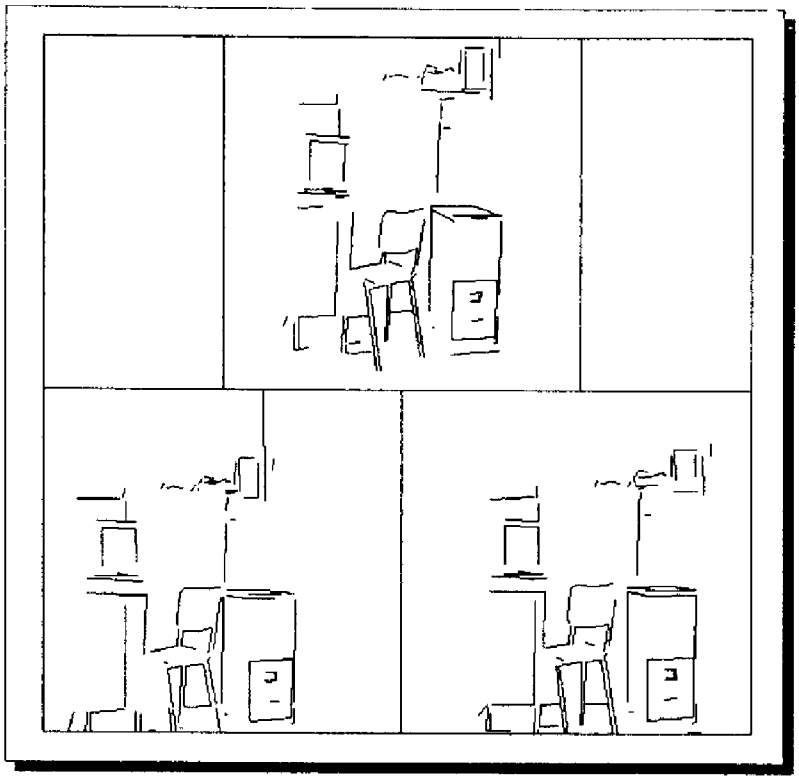


Figure IV.8: *Appariements stéréo initiaux de la scène*

à l'erreur issue de la stéréovision initiale, on voit que celle-ci décroit régulièrement en fonction du nombre d'observations.

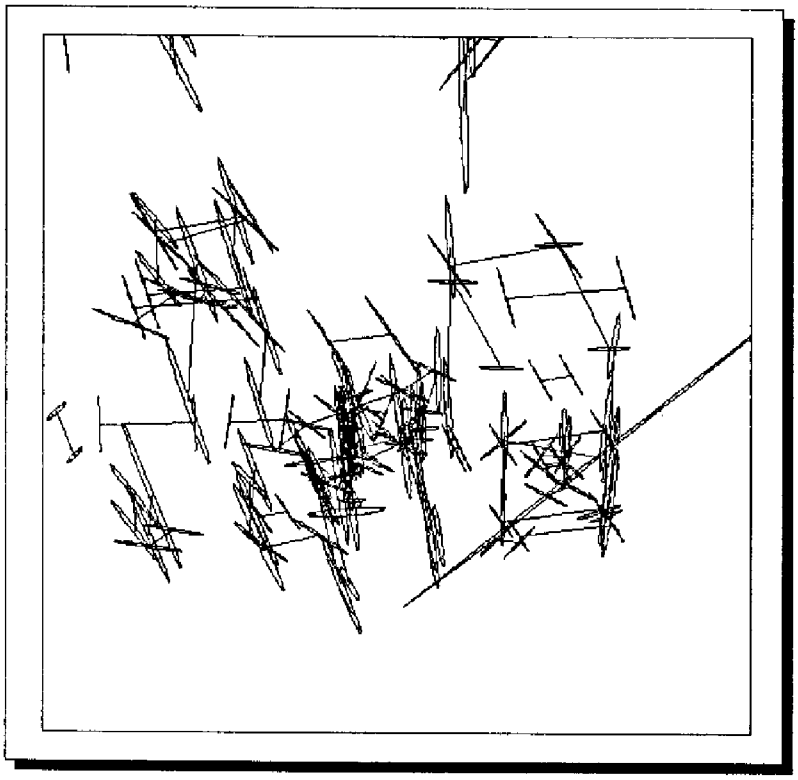


Figure IV.9: *L'image 3D originale (vue de 3/4 face) avec les ellipses d'incertitude associées aux extrémités des segments.*

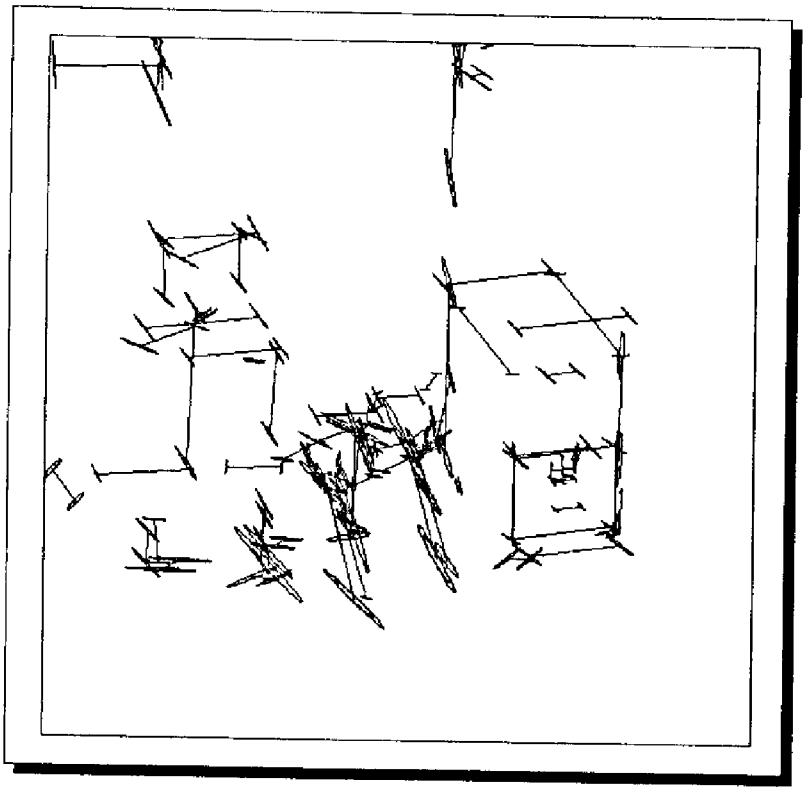


Figure IV.10: *L'image 3D après suivi (vue de 3/4 face)*

Numéro du suivi	Nombre de segments réestimés
1	68
2	62
3	38
4	44
5	47
6	46

Figure IV.11: *Nombre de segments suivis lors de chaque itération*

Nombre d'observations fusionnées	Nombre de segments concernés	Variance relative moyenne
0	106	1.00
1	90	0.47
2	78	0.31
3	58	0.22
4	45	0.17
5	31	0.15
6	12	0.11

Figure IV.12: *Evolution de la variance des droites 3D au cours de la fusion*

IV.5 Conclusion

Dans ce chapitre nous avons présenté un système de stéréovision dynamique qui apporte des solutions à certains inconvénients de la stéréovision classique : il permet d'obtenir un champ de vue plus large en déplaçant les caméras, il fournit une meilleure précision par fusion de plusieurs observations des segments 3D et il évite de calculer plusieurs fois les appariements de la stéréovision tout en présentant des possibilités supérieures de parallélisation.

La procédure de fusion 2D-3D mise en œuvre utilise un outil de filtrage général mis au point au LAAS qui permet d'intégrer facilement des informations en provenance d'autres capteurs : odométrie, télémètre laser... Cette formulation conduit néanmoins à des calculs plus lourds (la forme de l'équation de mesure n'est pas optimale. Le choix d'un autre repère assurerait une équation plus facilement linéarisable [Faugeras 89]).

Lors de la fusion, l'utilisation de la distance de Mahalanobis associée à notre équation de mesure pour vérifier la cohérence des appariements devrait permettre de réaliser une segmentation des segments suivis en ensembles de segments de mouvement homogène pour détecter des objets en mouvement dans la scène.

Chapitre V

La vision dans le système de contrôle d'un robot mobile

Dans ce chapitre, nous nous intéressons à l'intégration du système de perception dans un robot ; ce système doit être à même d'exécuter les fonctions vues précédemment aux chapitres II, III et IV.

Après une présentation rapide de la famille des robots Hilare qui servent de support à nos expérimentations, et de leur structure décisionnelle, nous verrons le formalisme utilisé pour la décomposition des différents modules qui les composent, et comment la vision s'y intègre.

V.1 Présentation d'Hilare 2

Le système de perception d'Hilare 2, le successeur d'Hilare, comporte :

- trois caméras formant un banc de stéréovision avec une base d'environ 60cm. Elles sont équipées d'objectifs de 12.5 mm de focale assurant un champ de vue d'environ 30 degrés. Elles sont montées sur une platine orientable en azimut.
- un télémètre laser orientable en site (miroir tournant) et en azimut (platine).
- trente-deux émetteurs-récepteurs à ultra-sons pour la proximétrie

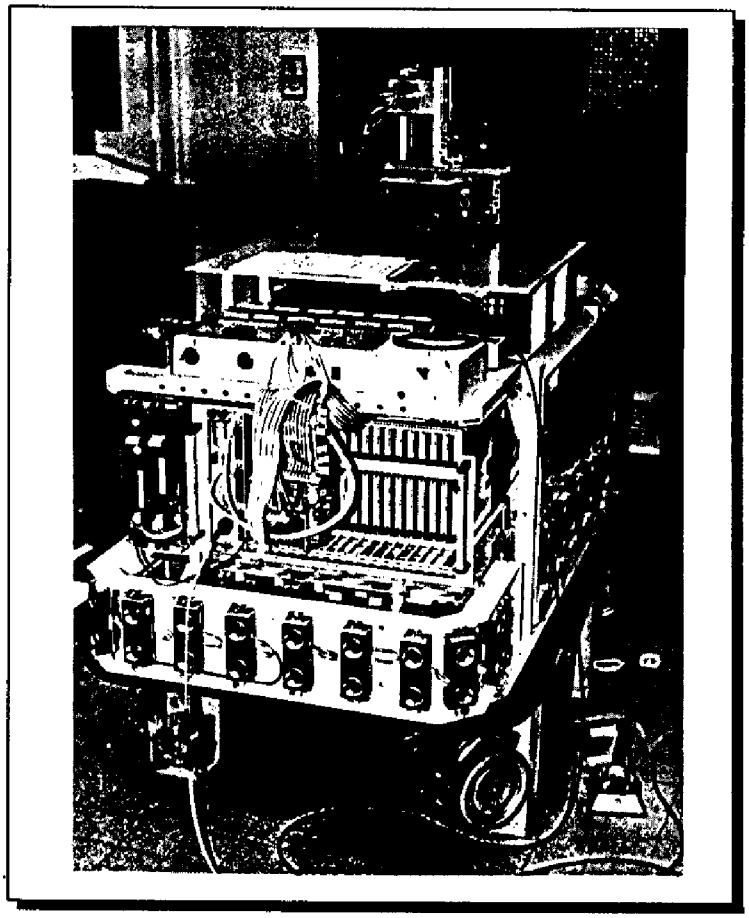


Figure V.1: *Le robot mobile Hilare 2*

- une nappe optique (laser) et une caméra associée pour la détection de vide (trous, marches...) et d'obstacles bas.
- un système de localisation proprioceptif comprenant un gyromètre, deux accéléromètres, un inclinomètre à deux axes, des roues odométriques couplées aux roues motrices et un odomètre vrai.
- un arceau pour la détection de contact.

Ce système de perception a pour rôle:

- La construction d'un modèle 3D de l'environnement en utilisant la stéréovision dynamique, la fusion des données acquises par le laser et la stéréovision,...

- La détection d'événements extérieurs nécessitant une certaine réactivité : obstacles, objets en mouvement, présence d'objets particuliers...
- Une contribution aux possibilités de contrôle des actions du mouvement du robot, par exemple la réalisation de mouvements asservis pour le suivi d'objet.

V.1.1 La structure informatique

L'infrastructure informatique embarquée comprend un nombre variable et extensible de cartes à microprocesseur supportant des logiciels de gestion, de traitement et de commande des capteurs, de construction et de manipulation des modèles de l'environnement, de calcul des déplacements, de commande et de contrôle des actionneurs (y compris à terme, une structure de manipulation — un ou deux bras). Deux chassis de 21 emplacements chacun sont prévus.

Pour les interactions avec l'utilisateur (au niveau tâche), le véhicule sera relié par radio (à 4800 bauds, par un protocole de type TCP/IP) à une station de travail Unix supportant l'interface utilisateur durant l'opération normale. Durant les périodes de développement et de mise au point, le robot sera lié par un câble Ethernet au réseau de stations de travail du laboratoire.

La structure logicielle d'Hilare 2 s'appuie sur la notion de modules coopérants. L'absence volontaire, en raison de la vocation expérimentale du robot, de spécifications rigides sur le flux d'information entre les modules conduit à une implantation dans laquelle deux modules quelconques peuvent a priori communiquer entre eux.

Concernant la structure matérielle embarquée (figure V.2), le choix s'est porté sur la famille Motorola : les cartes processeurs de la famille 32 bits $680xx$ et le bus VME comme moyen de communication. Ce choix est naturellement ouvert et permet l'intégration de processeurs spécialisés dédiés à des traitements particuliers. Notamment pour la vision, un ensemble de cartes Datacube et de Transputers assurera le traitement à bord.

La partie embarquée de la structure logicielle s'appuie sur les outils proposés par le système VxWorks : système d'exploitation temps réel (VRTX32), bibliothèques de services de communication inter-processeurs, moyens de développement croisé à partir des stations de travail, utilisation de langages évolués (C et

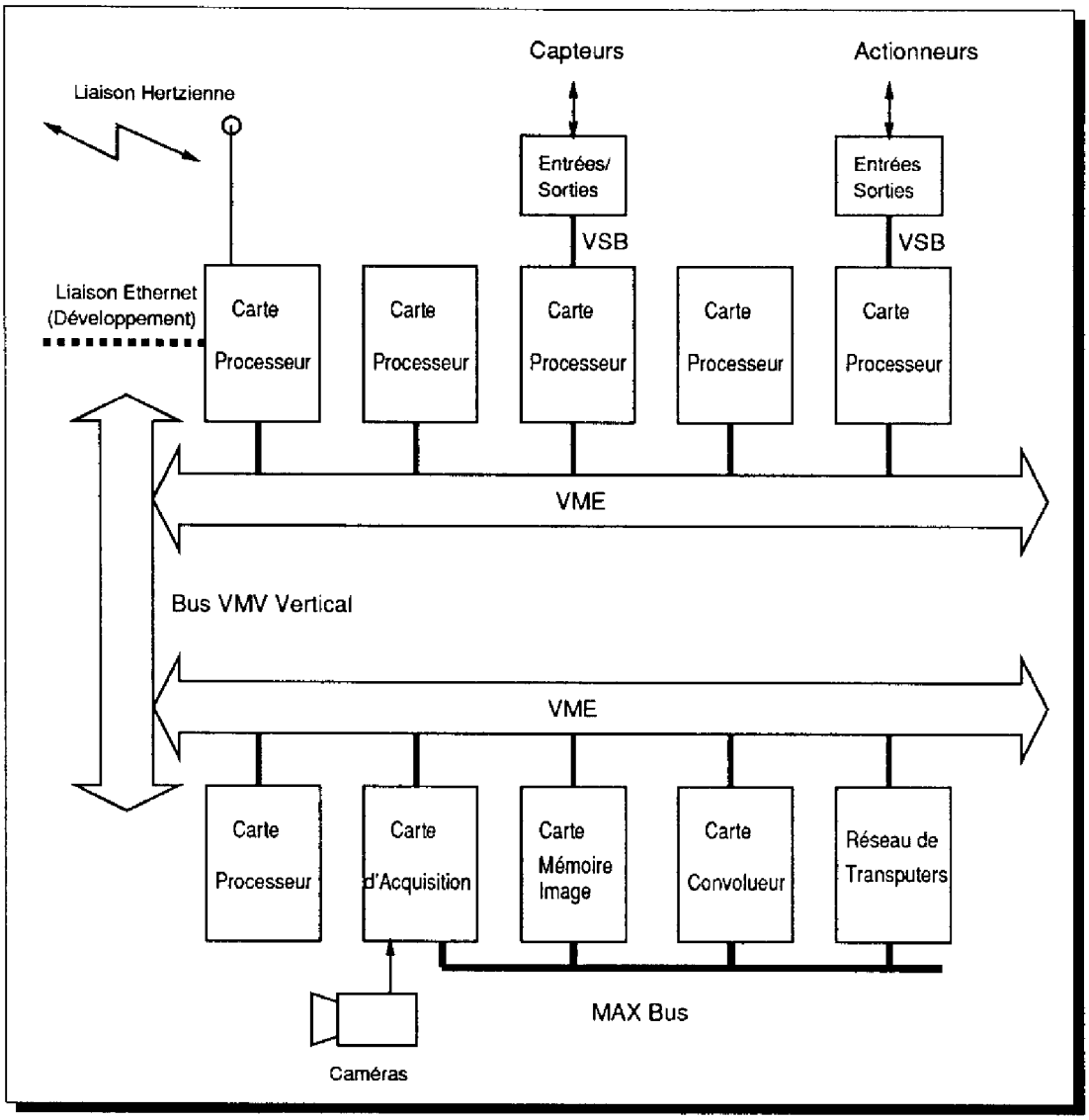


Figure V.2: *L'infrastructure informatique embarquée d'Hilare 2*

C++), débogueur symbolique.

Le bus VME et ses cartes processeurs sont vus depuis les stations de travail comme un sous-réseau Internet ; l'ensemble des services de communication classiques à ce niveau est disponible : TCP/IP, sockets, RPC, rlogins, NFS...

Néanmoins, pour des raisons d'efficacité, les processus «temps réel» embarqués n'utilisent pas le modèle Internet pour communiquer, mais des systèmes à base de boites aux lettres et d'affiches développés au LAAS [Ferraz 90, Chatila 90b].

V.2 La structure de contrôle d'Hilare 2

Un robot mobile doit planifier puis exécuter des actions. Le robot est donc constitué de deux entités : le système de planification et le système d'exécution, qui échangent des ordres et des comptes-rendus. Le système d'exécution doit néanmoins être doté de capacités de décision et de modifications locales du plan, pour permettre une réactivité face aux évènements non prévus dans le plan [Noreils 89b]. C'est pourquoi il est composé de deux parties : le système de contrôle et les modules fonctionnels.

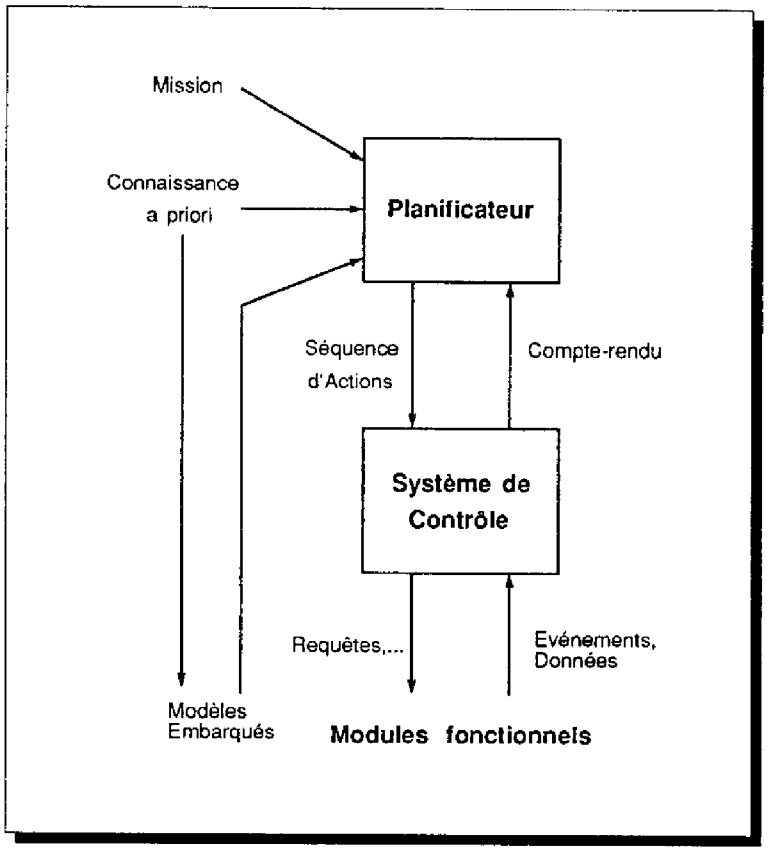


Figure V.3: *Les interactions entre le planificateur, le contrôle et les modules du robot.*

V.2.1 Les modules fonctionnels

Pour permettre aux systèmes de planification et de contrôle de raisonner sur les fonctions de base du robot, celles-ci sont regroupées en modules [Chatila 90b] et

classées selon leur type.

Deux modules communiquent par :

- envoi de requêtes
- envoi de compte-rendus d'exécution de requêtes
- envoi de contrôles
- envoi d'événements
- lecture ou écriture dans des entités partagées nommées *Structures de Données Exportées — SDE.*

Nous distinguons les quatre types suivants pour les fonctionnalités de base du robot :

Serveurs : Une telle fonction est déclenchée par une requête externe. Le résultat du traitement de cette fonction est placé dans une SDE qui sera lue par le module émetteur de la requête ou envoyé directement avec le compte-rendu d'exécution.

Filtres : Un filtre effectue un traitement systématique sur une SDE d'entrée, à une fréquence donnée, compatible avec la fréquence de rafraîchissement de la SDE d'entrée. Il peut produire une SDE en sortie, ou simplement modifier la SDE d'entrée. Le fonctionnement d'un filtre peut être interrompu de l'extérieur, mais ne s'arrête pas tout seul.

Asservissements : Un asservissement réalise une boucle fermée entre un module de perception et un module actionneur en utilisant une SDE fournie par la perception pour modifier une SDE d'entrée d'un module actionneur.

Surveillances : Les surveillances permettent de détecter des événements ou des situations internes ou externes au robot, et déclenchent la réaction du robot en envoyant un événement au système de contrôle. Selon le cas il peut s'agir de gestionnaires d'interruptions ou de démons qui testent à une fréquence donnée, l'occurence d'un événement particulier.

Chaque module du système de contrôle est constitué de fonctionnalités de base (serveurs, filtres, asservissements ou surveillances) et possèdent une ou plusieurs SDE en sortie. En entrée, ils reçoivent leurs données en provenance des SDE exportées par les autres modules. Les requêtes aux serveurs peuvent provenir soit de l'exécutif, soit d'autres modules.

A chaque module est associée une « tâche de contrôle » qui gère les requêtes et l'allocation des ressources internes au module, le lancement et l'arrêt des fonctions internes, ainsi que les modalités des différents algorithmes mis en oeuvre.

Pour limiter la complexité, toutes les SDE et les modules sont alloués statiquement lors de l'initialisation du robot. Les seules actions autorisées sont la mise en route ou l'arrêt de ces modules. La création dynamique n'est pas autorisée.

Pour permettre un parallélisme de type spatial sur les données, il est possible de créer plusieurs modules identiques, sur lesquels le système de contrôle répartira la charge de traitement lors de l'exécution.

Pour assurer le parallélisme de type fonctionnel (pipe-line), les SDE peuvent changer de propriétaire de manière dynamique pour permettre un passage rapide, de type mémoire commune, des données le long d'une chaîne de traitement. Ces SDE sont initialement attachées au système de contrôle, et lui sont retournées dès la fin du traitement.

Les *surveillances* jouent un rôle déterminant dans l'exécution de missions. Elles détectent l'apparition d'événements et déclenchent une action réflexe. Formellement une surveillance se présente sous la forme :

$$\texttt{MNTR}\ \textit{conditions} \Longrightarrow \textit{actions}$$

où *conditions* décrit l'événement (la condition sur les données du ou des capteurs que l'on surveille) et *actions*, les actions réflexes associées. On distingue deux types de surveillances : homogènes qui utilisent les données d'un seul capteur et hétérogènes qui combinent, numériquement ou sémantiquement, les données de plusieurs capteurs ; ce dernier type de surveillance sera implémenté dans un module séparé et émettra des requêtes vers les modules capteurs à surveiller.

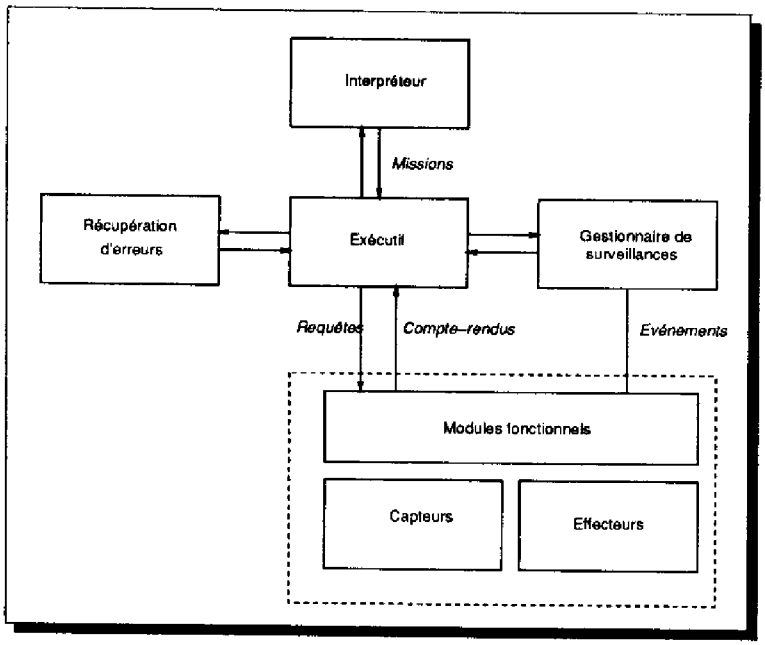


Figure V.4: *L'architecture du système de contrôle d'exécution*

V.2.2 Le système de contrôle

Cette entité réalise l'exécution des missions en provenance des niveaux supérieurs. Compte tenu de la mission à exécuter, il s'assure de la disponibilité des ressources nécessaires et émet des requêtes vers les modules concernés. Il gère les compte-rendus fournis, et active un système de récupération d'erreur en cas d'échec (figure V.4). La thèse de F. Noreils contient une description plus complète de du système de contrôle et de récupération d'erreurs implémenté sur Hilare 1 et Hilare 1.5 [Noreils 89a].

De plus, le système de contrôle intègre des mécanismes de communication et de synchronisation qui permettent son utilisation dans un environnement multi-robots [Noreils 90a].

V.3 Intégration de modules de perception

Nous décrivons ici comment les fonctions de perception s'intègrent dans l'architecture que nous avons présentée.

V.3.1 Les fonctions de perception

Le système de perception embarqué d'un robot mobile doit fournir les fonctionnalités suivantes :

- Modélisation 3D de l'environnement du robot en utilisant les différents capteurs disponibles. Le modèle est construit en utilisant divers types de primitives tri-dimensionnelles telles que facettes planes, segments de droite dans l'espace 3D, ou 2.5-dimensionnelles (carte d'élévation du terrain).

  Ce modèle est utilisé par le contrôle pour :

  - calculer un modèle de l'espace libre pour la génération de chemins et pour abstraire des représentations topologiques et sémantiques.
  - identifier certains éléments connus, tels que les portes, les couloirs ou des objets connus,
  - localiser le robot en fusionnant la vue courante avec le modèle existant (procédure de recalage-fusion décrite dans [Moutarlier 89c]).

- Aide à la navigation fournissant des fonctions du type asservissement liant des données capteurs (position d'un objet, forme d'un obstacle,...) aux commandes des actionneurs des roues.

- Surveillances liées aux données perceptuelles : détection d'obstacles, reconnaissance d'un endroit,...

- Interaction avec les niveaux supérieurs du système, par exemple un opérateur, pour d'une part présenter une vue synthétique de la connnaissance courante du robot ou des connaissances symboliques extraites des données sensorielles et d'autre part acquérir des connaissances a priori telles que des modèles d'objets ou des plans de pièces.

- Planification interne des tâches de vision : utilisation des capteurs, choix de points de vue, des algorithmes en fonction des tâches à réaliser [Colly 89].

Du point de vue de l'architecture du système de contrôle, ces fonctions peuvent être classées en quatre types d'activités, qui correspondent aux types fonctionnels définis plus haut [Chatila 90a, Devy 90] :

fonctions de perception temps-réel : ces fonctions sont utilisées dans la boucle fermée de commande du robot. Ce sont des asservissements ou des filtres.

fonctions de surveillance : elles sont activées que le robot soit en mouvement ou non. Ce sont des surveillances. Ces fonctions ont une fréquence de calcul fixe qui doit être respectée pour assurer la sécurité du robot.

fonctions de modélisation : ce sont les fonctions qui entrent en jeu dans la boucle de modélisation de l'environnement :

1. perception (acquisition de données).
2. modélisation locale
3. fusion avec les connaissances déjà acquises
4. calcul de point de vue

Ces fonctions ne sont pas nécessairement temps réel, même si elles conditionnent la vitesse d'évolution du robot. De plus, elles peuvent être effectuées en parallèle avec d'autres tâches, notamment le déplacement du robot ; certains affinements du modèle peuvent être faits sur requête, de manière asynchrone.

fonctions asynchrones : il s'agit de serveurs devant satisfaire de nombreuses requêtes externes qui peuvent être traitées en temps masqué, en parallèle avec d'autres activités : consultation du modèle (par l'opérateur ou le planificateur), modification des modalités d'un capteur ou d'un algorithme, activation d'une nouvelle fonction...

Ces requêtes permettent aux autres modules d'interagir avec le système de perception.

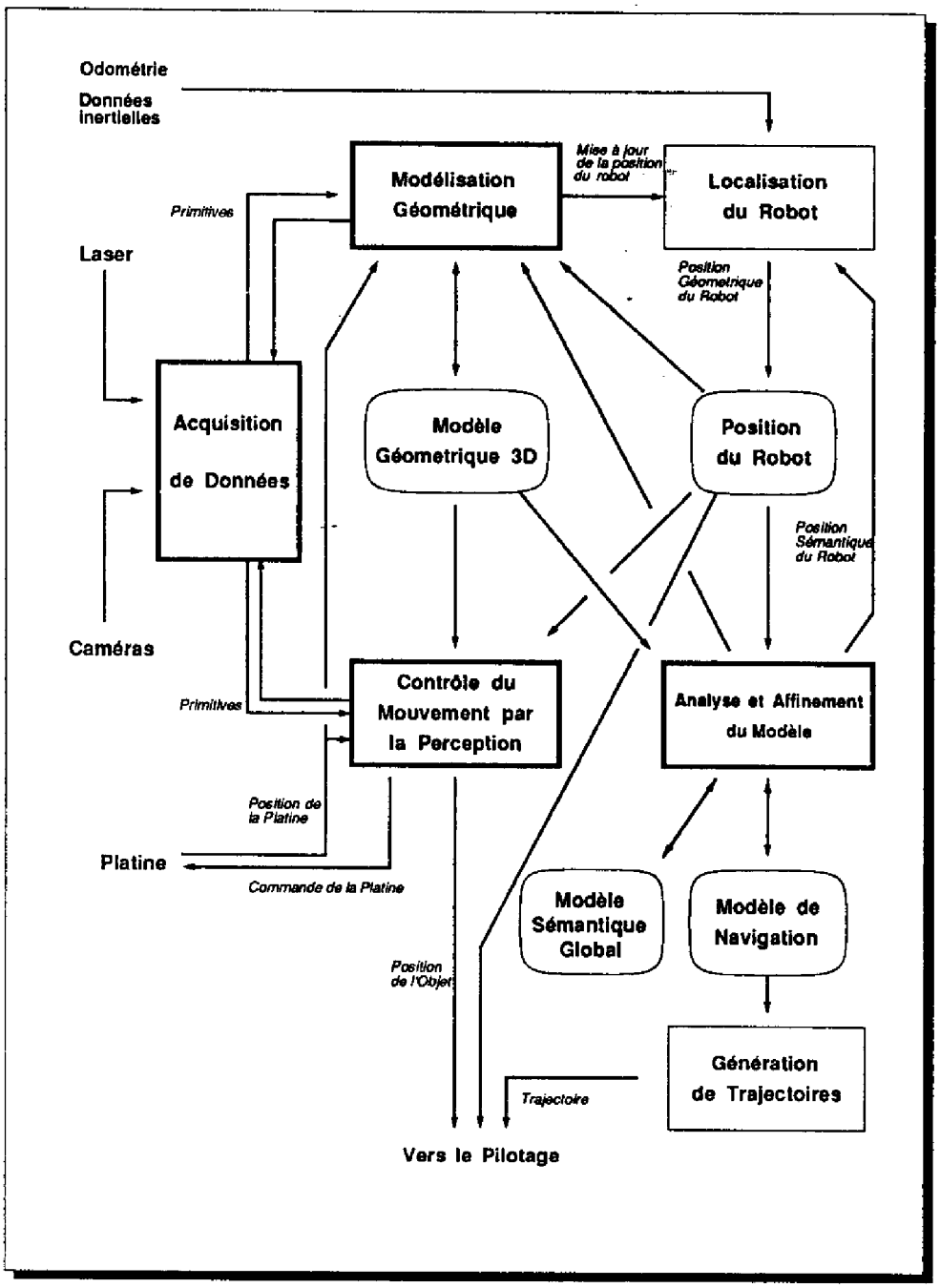


Figure V.5: *L'architecture du système de perception. Les fonctions sont représentées par des boites à coins carrés et les SDE par des boites à coins arrondis*

La figure V.5 présente d'un point de vue général l'architecture du système de perception d'Hilare 2, en mettant en évidence les relations entre les SDE et les fonctions.

On trouve les fonctions suivantes, dont certaines seront détaillées dans la suite :

- Acquisition de données et extraction de caractéristiques.
- Modélisation géométrique incluant la construction d'un modèle 3D local et son agrégation au modèle 3D global.
- Analyse et raffinement du modèle (aux niveaux topologiques et sémantiques) pour obtenir un modèle de navigation et un modèle sémantique et topologique global.
- Contrôle du mouvement à partir de la perception pour l'exécution de portions de trajectoires à partir du modèle de navigation (suivi d'objets, de route,...).
- Localisation du robot. Cette fonction est responsable du calcul et du maintien à jour de la meilleure estimation possible de la position du robot.

Nous allons maintenant examiner en détail les modules qui font plus particulièrement intervenir la vision : l'acquisition de données, le contrôle du mouvement à partir de la vision, et l'acquisition de données 3D par la stéréo dynamique.

V.3.2 Acquisition de données

L'acquisition des données est réalisée par un ensemble de modules qui sont liés directement aux capteurs (caméras, télémètre laser...) et qui fournissent des données capteur traitées (primitives issues de la segmentation, extraction de points caractéristiques...).

L'acquisition des images sur Hilare 2 est faite par l'intermédiaire des cartes Datacube DIGIMAX et ROI-STORE. Les caméras ne sont pas commandables (mise au point et diaphragme fixes). Les données de contrôle pour chaque caméra sont :

- Les paramètres de la numérisation (gain, offset et type de filtrage)

- L'identificateur de la SDE où stocker l'image

L'orientation des trois caméras en azimut peut être contrôlée par rotation de la platine (envoi d'une requête au module de commande de la platine).

Les traitements de base, dans notre cas essentiellement l'extraction des segments, sont réalisés sur requête par un *serveur* qui lit une SDE contenant l'image brute et produit une nouvelle SDE contenant les segments extraits. Le processus d'extraction des segments a été vu au chapitre II. Les données de contrôle de ce module sont :

- La région d'intérêt de l'image. Les segments ne sont extraits que dans cette région.
- Les paramètres des différents algorithmes mis en œuvre : seuils pour l'extraction des points de contour, erreur maximale sur l'approximation polygonale, longueur minimale des segments.

Les autres modules d'acquisition (laser [Briot 90], ultra-sons, odométrie [Bauzil 88]) ne seront pas décrits ici.

V.3.3 Contrôle du mouvement par la perception

Le suivi d'objet décrit au chapitre III est un exemple de l'utilisation de la vision pour la navigation du robot : la mission du robot est de détecter puis de suivre un objet particulier dont le modèle 3D de type CAO est connu. L'exécution de cette mission consiste à :

1. mettre en œuvre une surveillance dont le déclenchement signifie que l'objet a été détecté,
2. localiser l'objet,
3. le suivre.

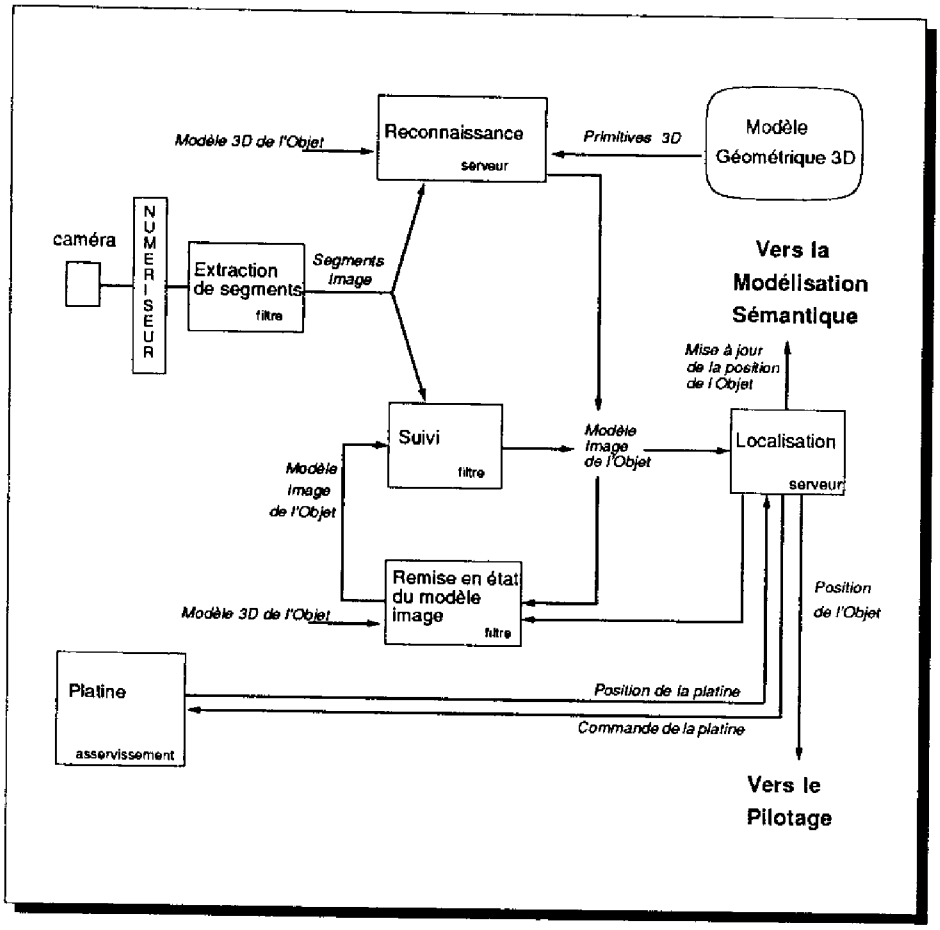


Figure V.6: *Les fonctions du module de suivi d'objet pour la vision*

Détection de l'objet

Dans notre système, le module de reconnaissance présenté au chapitre III est un *serveur* qui utilise le modèle CAO de l'objet et les segments image pour produire le modèle image de l'objet.

Les requêtes vers ce module sont produites par le gestionnaire de surveillances qui gère la surveillance sur la présence de l'objet particulier dans l'environnement. La figure V.7 montre le programme de gestion de cette surveillance tel qu'il est écrit sur Hilare. Le formalisme utilisé est décrit dans [Noreils 89a].

```
# surveillance to detect the object
surv-one : MNTR (S_detect-visual-object)=>(){
    (set exit-var (0))
}
#we are seeking for the object
(while (exit-var)(
    (turn-platform (0))
    (vision-comp) # request to vision module
    (inc-platform := inc-platform + 0)
    (if (inc-platform > 360)
        ((add-to-historic "pattern not found")(FAIL))
        ())
))
```

Figure V.7: *La déclaration de la surveillance sur la détection de l'objet. Une boucle* `while` *réalise une exploration de l'environnement et émet des requêtes vers la vision. Lorsque la vision a reconnu l'objet (*`S_detect_visual_object`*), la surveillance devient vraie et la boucle se termine.*

Localisation

La fonction de localisation présentée au paragraphe III.3.1 est implémentée comme un *serveur* qui reçoit des requêtes de la fonction de reconnaissance d'une part et de la fonction de suivi de l'autre. Ses entrées sont le modèle CAO, le modèle image de l'objet et les matrices de calibrage de la caméra. En sortie, il fournit la matrice de projection représentant la localisation de l'objet, associée à la valeur de l'erreur quadratique moyenne (score de qualité). Il utilise un seul élément de contrôle : le seuil de qualité sur la localisation utilisé pour la détection de mauvais appariements.

La position de l'objet peut être, si nécessaire, enregistrée ou mise à jour dans le modèle sémantique du monde.

Suivi

Le suivi est un *filtre* qui prend en entrée un modèle image de l'objet et les segments extraits d'une nouvelle image et fournit le nouveau modèle image. Ses contrôles sont les valeurs de l'incertitude sur la prédiction des segments (région de recherche) et les différents seuils de validation pour la recherche des cliques maxi-

males dans le graphe de compatibilité.

Remise en état du modèle image

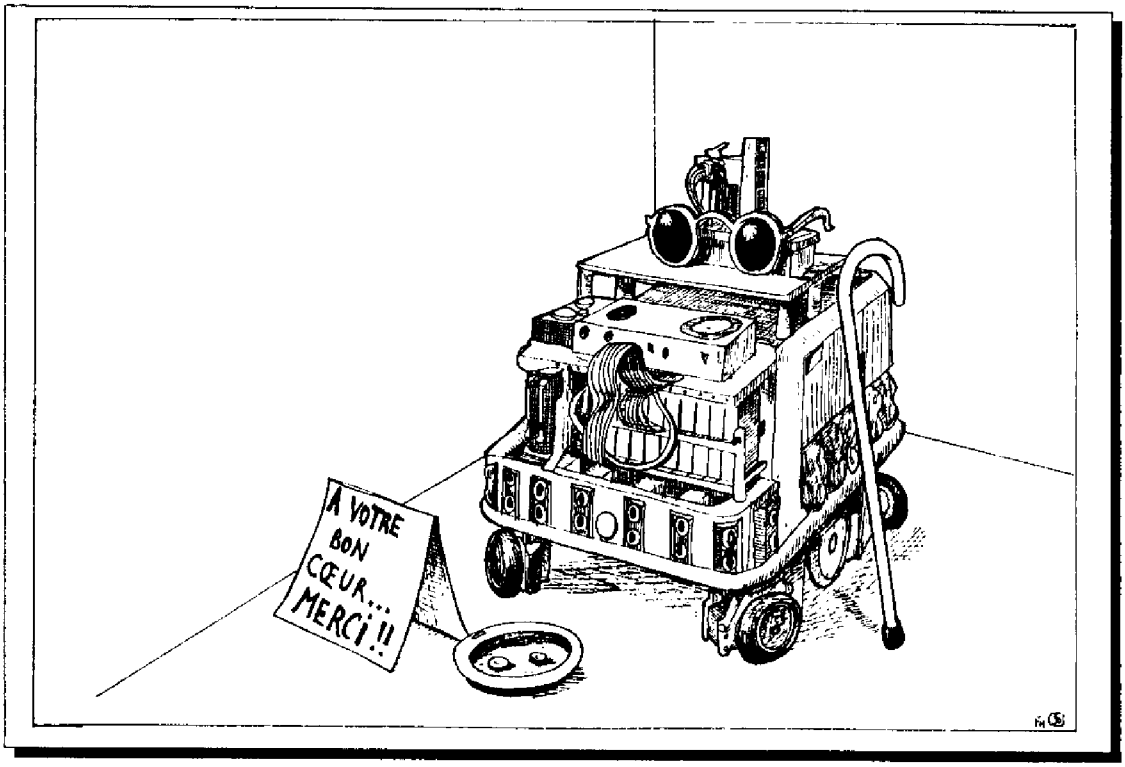
En cas de faux appariements ou de perte de segments, la remise en état du modèle objet est réalisée par un *filtre* qui, en utilisant la localisation de l'objet, les données de calibrage de la caméra et le modèle CAO de l'objet, fournit un nouveau modèle image de l'objet complet.

V.3.4 Acquisition de données par stéréovision dynamique

Le module de suivi tri-caméra présenté au chapitre IV est utilisé pour fournir les données 3D. Cette fonction est vue comme un *serveur* qui répond aux requêtes du superviseur. C'est notamment sur ordre du superviseur qu'une nouvelle stéréovision sera déclenchée sur les segments non appariés (voir figure IV.6, page 75).

Ce module lit et écrit une SDE contenant le modèle en cours de construction. Lors de chaque nouvelle fusion, il lit la SDE qui contient la position du robot mesurée par les différents capteurs proprioceptifs et propose une nouvelle valeur au gestionnaire de la localisation du robot (voir figure V.5). Il obtient les nouveaux segments image du module d'acquisition décrit précédemment.

Les contrôles de ce module sont très nombreux: les différents paramètres des méthodes numériques qui interviennent (linéarisation, différentiation, filtre de Kalman...).


A VOTRE
BON
CŒUR...
MERCI !!

V.4 Repères et calibrages

Un problème clé de la robotique est le calibrage des positions des repères locaux liés aux différents capteurs et effecteurs. Nous présentons ici les solutions apportées dans le cas des robots de la famille Hilare.

V.4.1 Le graphe des repères d'Hilare

La figure V.8 montre le graphe qui lie les différents repères d'Hilare, pour l'exécution d'un mouvement asservi sur un objet.

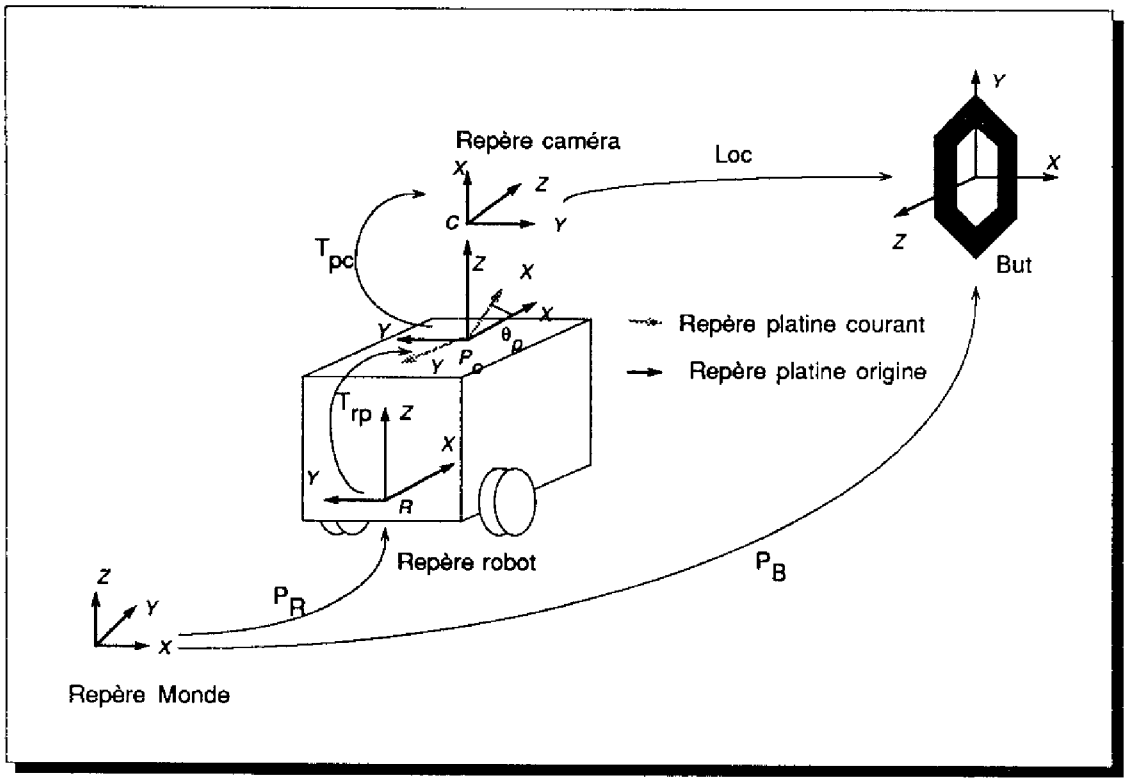


Figure V.8: *Le graphe des repères d'Hilare*

On y trouve les repères suivants :

le repère robot $\mathcal{R}$: lié au centre de l'axe des roues et orienté perpendiculairement à celui-ci. Son déplacement est mesuré par l'odomètre.

le repère platine origine $\mathcal{P}_O$: lié à l'axe de rotation de la platine et à sa direction initiale (0).

le repère platine courant $\mathcal{P}$: qui tourne avec la platine.

le repère stéréovision $\mathcal{C}$: lié au système de vision et déterminé lors de son calibrage (les matrices perspectives des trois caméras sont exprimées dans le même repère).

$\mathcal{R}$ est relié à $\mathcal{P}_\mathcal{O}$ par la transformation homogène T_{rp}, $\mathcal{P}_\mathcal{O}$ est relié à $\mathcal{P}$ par T_{pp} qui est simplement la rotation d'angle θ_P (position angulaire de la platine) autour de l'axe vertical P_z et $\mathcal{P}$ est relié à $\mathcal{C}$ par T_{pc}.

P_R et P_B sont les positions du repère Robot et du repère Objet dans le repère global exprimées sous forme de transformations.

Un module spécialisé prend en charge l'ensemble des transformations nécessaires entre ces repères. Chaque module d'acquisition ou de commande exprime ses données dans son propre repère local. Le contrôle émet lui-même les requêtes pour les transformations qui doivent être calculées.

Dans le cas du module de modélisation, les transformations sont calculées en prenant en compte explicitement les incertitudes de manière à pouvoir utiliser les outils généraux de fusion numérique (filtre de Kalman étendu).

V.4.2 Calibrage Robot-Platine

La méthode de calibrage utilisée est décrite en détail dans [Grandjean 89b]. Il s'agit de déterminer la liaison rigide T_{rp} entre $\mathcal{R}$ et $\mathcal{P}_\mathcal{O}$. Dans le cas d'Hilare nous supposons que l'axe de rotation de la platine est perpendiculaire au plan dans lequel se déplace le robot, $\mathcal{R}_{xy}$. T_{rp} est donc considérée comme une transformation plane[1] (une translation composée avec une rotation autour de l'axe vertical z).

Pour la déterminer, nous utilisons deux mesures des déplacements du robot :

- l'odométrie qui mesure les déplacements de $\mathcal{R}$: D_R
- le système de recalage utilisant le télémètre laser [Moutarlier 89a] qui mesure les déplacements de $\mathcal{P}_\mathcal{O}$: D_P.

[1] Cette hypothèse est liée à l'impossibilité de faire effectuer au robot des mouvements hors de ce plan pour mesurer les autres composantes de la transformation

La rigidité du robot nous donne alors l'équation :

$$T_{rp}.D_P = D_R.T_{rp}$$

En faisant effectuer au robot une séquence de mouvements, on obtient un ensemble de mesures qui permettent d'estimer T_{rp} itérativement par un filtre de Kalman étendu.

L'avantage de cette méthode est qu'elle ne nécessite aucun dispositif particulier, et qu'elle peut donc être réalisée automatiquement.

V.4.3 Calibrage Platine-Caméras

La méthode utilisée a été développée par P. Grandjean et est décrite dans [Grandjean 89a]. Elle utilise la fusion entre les données 3D issues de la stéréovision et la coupe plane de l'environnement réalisée par le télémètre laser. La transformation est estimée itérativement par filtrage de Kalman étendu, a partir des correspondances entre les segments 3D et les mesures du laser. L'équation de mesure de ce filtre exprime que le point mesuré par le laser appartient au segment correspondant localisé par la stéréovision dans le repère des caméras.

Cette méthode est très souple et facile à mettre en œuvre, car elle ne nécessite pas la connaissance exacte de points d'une mire, et sa précision peut devenir très bonne en accumulant un grand nombre de correspondances sur plusieurs images différentes.

V.5 Expérimentations

En collaboration avec Fabrice Noreils, des expérimentations ont été menées avec les robots mobiles Hilare et Hilare 1.5 sur le suivi d'objets. Actuellement, une partie des logiciels de contrôle et tout le logiciel de vision sont déportés sur les stations Sun. Les images vidéo sont transmises par câble aux cartes d'acquisition et de traitement Datacube, installées dans le chassis d'une station Sun.

Nous avons réalisé plusieurs types d'expériences montrant les capacités conjointes du système de contrôle et du suivi par la vision [Noreils 89a] :

1. l'objet est fixe, et la mission principale du robot est de le rejoindre. Durant son mouvement, le robot évite les obstacles imprévus (figure V.9). Les capacités de reprise sur erreur du système permettent de prendre en compte de brusques changements de position de l'objet (figure V.10).

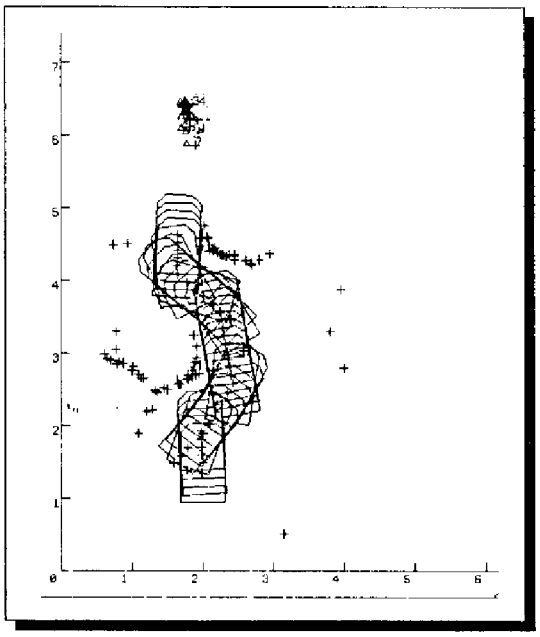


Figure V.9: *Le robot se dirige vers l'objet en évitant deux obstacles.*

2. l'objet est mobile. Il est fixé à l'arrière du robot Hilare 1 et le système de contrôle multi-robot est utilisé pour synchroniser les déplacements. Hilare 1 effectue une mission (recherche puis suivi de paroi) et Hilare 1.5 le suit (figure V.11). La coopération multi-robots est utilisée pour récupérer l'erreur qui se produit lors de la disparition d'Hilare derrière le coin.

Dans tous les cas, le fait que la localisation soit d'autant plus précise que l'on est près de l'objet se manifeste par la répartition des croix montrant le but : c'est un nuage qui comporte un fort point d'accumulation autour de la position réelle. En effet plus le robot se rapproche, plus l'erreur diminue.

Dans l'état actuel de l'implémentation, la période de l'ensemble (intervalle de temps entre deux prises d'images) est de 2 à 3 secondes. Ce temps inclue l'extraction des segments, le suivi et la localisation de l'objet, le mouvement de la platine puis le mouvement du robot.

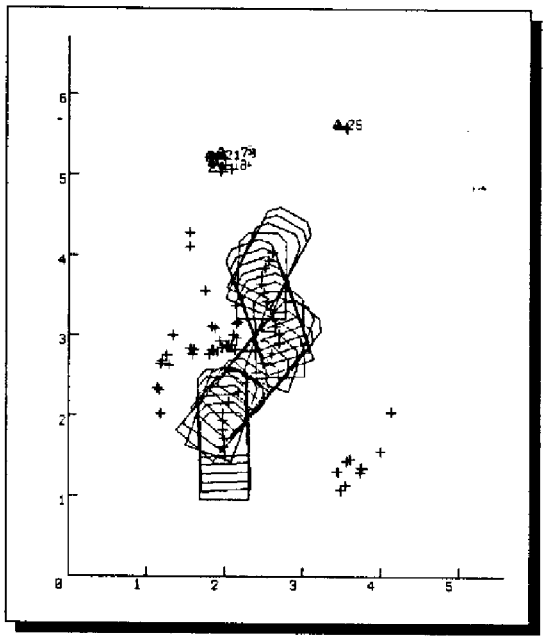


Figure V.10: *Le robot se dirige vers l'objet qui change de position au cours de l'évitement d'obstacle. Le système de récupération d'erreur permet de retrouver l'objet.*

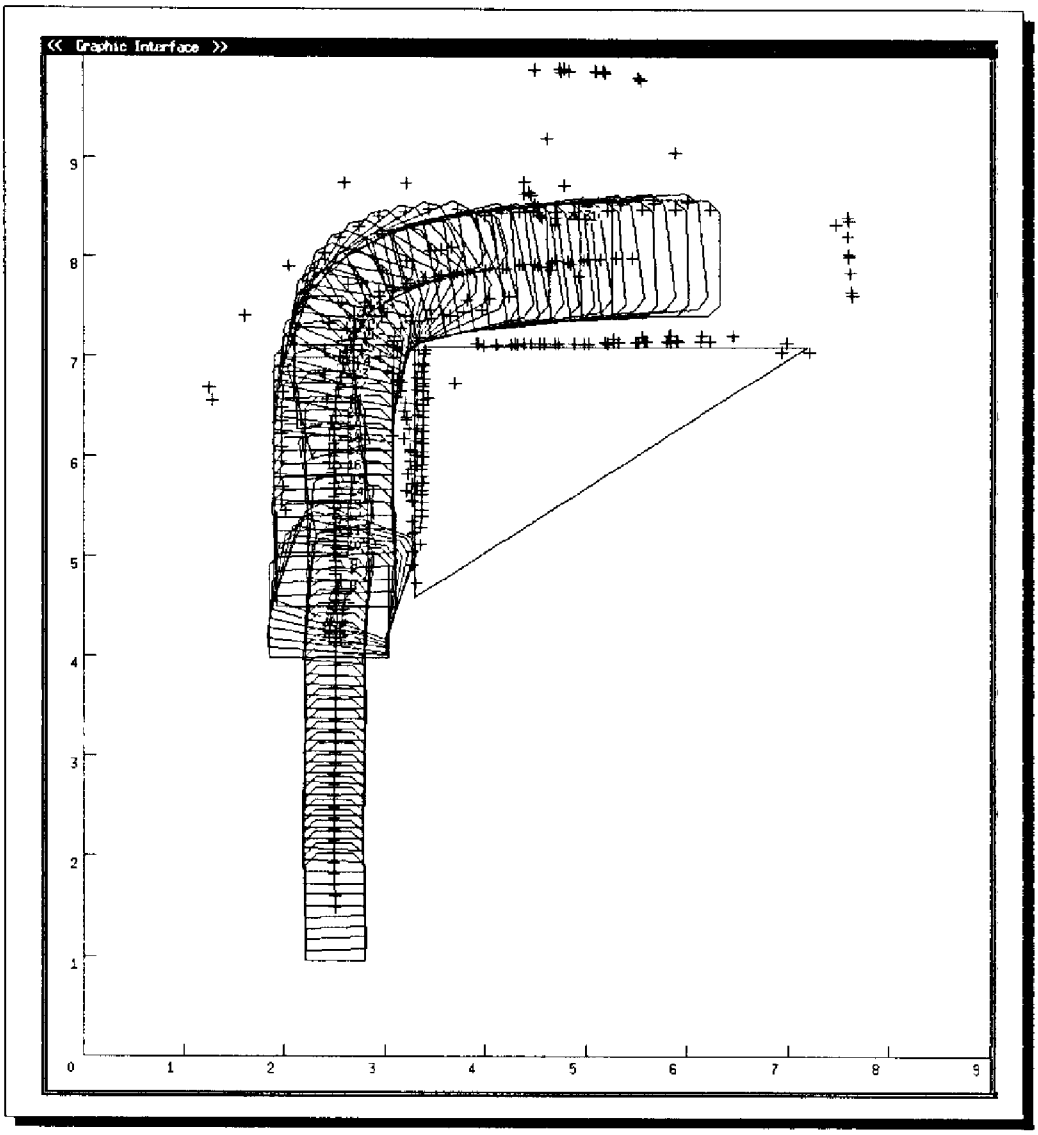


Figure V.11: *Suivi du robot Hilare 1 avec coopération multi-robots : lorsque l'objet est perdu après que Hilare 1 ait tourné le coin, Hilare 1.5 suit la paroi pour retrouver Hilare 1 et le localise à nouveau.*

V.6 Conclusion

L'analyse des différentes fonctions que peut réaliser la vision pour le système de contrôle du robot mobile et la classification des fonctions de base du robot en fonction de leur type d'activité ont permis de définir formellement les modules fonctionnels qui font intervenir la vision au sein de l'ensemble de l'architecture du système de perception.

Nous avons ensuite passé en revue les différents calibrages nécessaires entre les capteurs de nature différente et les effecteurs pour permettre d'une part la réaction en boucle fermée sur les capteurs et d'autre part la fusion multi-sensorielle.

Les expérimentations réalisées sur Hilare 1.5 en utilisant le système de suivi d'objet présenté au chapitre III démontrent le bon fonctionnement de cette structure de contrôle.

La réalisation d'Hilare 2 va permettre d'intégrer au sein du système embarqué l'ensemble des fonctionnalités existantes qui fonctionnent actuellement de manière séparée.

Conclusion générale

Nous avons étudié dans cette thèse divers problèmes liés à la perception en mouvement par vision artificielle pour les robots autonomes.

Nous pouvons classer les difficultés posées par ce problème en deux catégories :

- L'algorithmique propre liée au suivi des caractéristiques dans les images, à l'analyse du mouvement observé et à l'extraction de données tridimensionnelles.
- Les aspects «système» liés aux applications visées ; d'une part des problèmes de temps d'exécution dus en grande partie au volume important des données à traiter et aux contraintes de réactivité du robot, et d'autre part des problèmes d'architecture pour lier effectivement les capteurs et les effecteurs à l'intérieur du système de contrôle du robot.

Ce n'est pas entièrement par hasard que les deux aspects système du problème encadrent les aspects algorithmiques dans notre exposé ; les progrès du matériel qui ont permis la réalisation de systèmes informatiques performants et bien adaptés au traitement d'images ont relancé l'intérêt pour l'utilisation du mouvement en vision artificielle appliquée à la robotique : ces nouveaux moyens permettent de fermer de manière de plus en plus serrée la boucle perception—décision—action.

Ce regain d'activité a entraîné un besoin nouveau d'intégration des systèmes de vision artificielle et des logiciels de contrôle des robots. Contrôler la perception de manière globale et performante n'est pas évident et pour réaliser de véritables

boucles de commande sur la vision, il faut plus qu'une ligne série entre une machine de vision et une armoire de commande.

Au niveau algorithmique, le mouvement enrichit beaucoup la perception d'un robot. En effet, l'information apportée est de deux natures : d'une part il y a l'information différentielle sur l'environnement, et de l'autre il y a un élargissement du champ de vue : voir en mouvement permet de réaliser des panoramiques, des travelings, ou des effets de zoom qui apportent la même richesse au modèle que ces effets de caméra en apportent au spectateur de cinéma. Enfin il y a l'aspect agrégation qui vient du fait qu'en voyant les mêmes éléments plusieurs fois on augmente la précision de leur localisation.

Les travaux présentés dans ce mémoire traitent des problèmes appartenant aux deux catégories. Le système d'extraction de segments de contours dans les images présenté au chapitre II sera intégré à moyen terme dans le robot Hilare 2. L'utilisation conjointe du parallélisme et des architectures spécialisées permet d'atteindre des performances compatibles avec les contraintes du « temps réel » sur un robot mobile.

Le système de suivi de caractéristiques que nous avons présenté au chapitre III a montré son efficacité et sa fiabilité dans les deux applications présentées aux chapitres III et IV. Le suivi d'objet implémente un mode de contrôle du mouvement qui est très utile dans le cadre de missions d'inspection ou de surveillance.

La stéréovision dynamique présentée au chapitre IV est un moyen puissant d'acquérir une information tridimensionnelle riche et précise sur l'environnement du robot. Son utilisation par les systèmes de modélisation incrémentale ou de reconnaissance, en collaboration avec les données en provenance d'autres capteurs, permettra de combler les espaces laissés par les systèmes actuels qui donnent plusieurs vues statiques difficiles à fusionner.

Enfin, la prise en compte explicite de la perception dans une architecture de contrôle du robot, telle que celle qui a été conçue pour Hilare 2 (chapitre V) est nécessaire pour permettre la réalisation de missions utilisant la perception de manière active : contrôle des déplacements en boucle fermée par la vision, choix de points de vue, utilisation des modèle fourni par les modules de perception pour la

planification des trajectoires et leur exécution.

Perspectives

La réalisation, sur le robot mobile Hilare 2, d'un système de perception réellement intégré au robot et incluant des cartes spécialisées et un réseau de Transputers permettra de tirer plein bénéfices des travaux déjà menés dans ce domaine, et ouvre la voie à l'implémentation parallèle des algorithmes de plus haut niveau qui n'a pas été évoquée ici.

Du coté de l'algorithmique, plusieurs problèmes restent ouverts : le suivi d'objets réellement tridimensionnels, en prenant en compte les changements progressifs d'aspect, offre des perspectives intéressantes notamment pour des problèmes de manipulation de ces objets, ou même pour permettre à deux robots de se suivre sans monter une cible particulière sur le premier.

De plus, il est envisageable de suivre visuellement et de déterminer le mouvement d'un objet mobile inconnu de la base de donnée du robot.

Un autre domaine ouvert est celui de la segmentation du mouvement : actuellement on suppose en général que la scène observée est rigide : soit on ne s'intéresse qu'à un seul objet en mouvement en ignorant le fond (chapitre III), soit on considère des caméras en mouvement dans un environnement statique (chapitre IV). En réalité il est important de pouvoir travailler sur plusieurs mobiles : en reconnaissant les ensembles de mouvement homogène d'abord, puis en intégrant cette information numérique et sémantique dans les modèles.

Au niveau de la prise en charge de la perception par le système de planification et de contrôle des robots, beaucoup de choses restent à étudier : planification des déplacements pour optimiser les points de vue (génération de « plans de perception »), contrôle des modalités des capteurs en fonction des conditions extérieures...

Enfin, un aspect capital qui n'a pas été abordé ici est celui de l'utilisation des données fournies : la construction d'un modèle de l'environnement basé sur des primitives plus riches : plans, volumes, étiquettes sémantiques...

La réalisation de robots réellement autonomes et programmables reste un objectif scientifique passionant. La perception est une capacité fondamentalement nécessaire à de telles entités. L'utilisation du mouvement permet d'enrichir cette perception. Il reste à réaliser des robots tels que Hilare 2 sur lesquels ces capacités seront réellement toutes intégrées et qui réalisent des missions hors des laboratoires : dans un environnement réel, sur terre ou ailleurs.

Annexe A

Calcul du gradient sur cartes Datacube

On souhaite calculer, en utilisant les cartes Datacube, le module et la phase du gradient d'une image. Les composantes en x et en y du gradient sont d'abord calculées par convolution de l'image initiale avec un masque de Sobel, puis le module et la phase sont calculés à partir des valeurs obtenues.

On utilise pour cela trois cartes de la gamme Datacube : une mémoire image FRAME-STORE, une carte convolueur VFIR et une carte UAL d'images MAX-SP. Les connexions effectuées sur MAX-BUS entre ces cartes sont indiquées sur la figure A.1. Pour des raisons de clarté, il n'a été représenté que les sous-ensembles des cartes utilisés réellement dans le calcul.

A.1 Calcul du gradient en x et y

La dérivée de l'image dans une direction (x ou y) est calculée par convolution de l'image avec un masque de Sobel. Cette opération est réalisée en utilisant la carte VFIR en une trame vidéo.

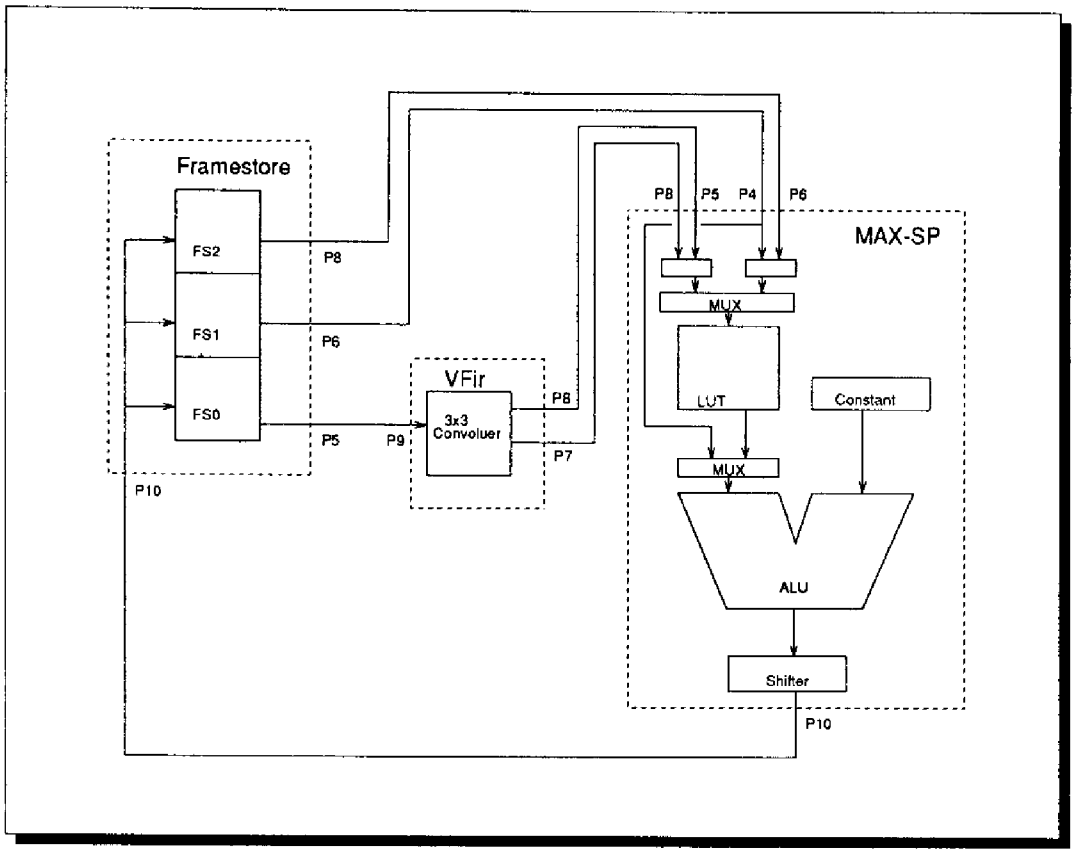


Figure A.1: *Les connexions possibles entre les unités de traitement des cartes*

A.1.1 Limitation de l'amplitude de l'entrée

Le masque de Sobel étant composé de coefficients positifs et négatifs, la carte VFIR doit être configurée pour travailler en mode *complément à 2*. Or l'image d'entrée étant par défaut non signée (elle prend ses valeurs entre 0 et 255), il faut soit la diviser par 2, soit limiter le gain de la numérisation pour pouvoir la considérer comme signée.

Dans le cas où l'image originale comprend des valeurs entre 0 et 255, la carte MAX-SP est utilisée pour réduire cette amplitude par le registre à décalage en sortie de l'unité arithmétique (qui est configurée pour additionner zéro à l'entrée P4 par laquelle arrive l'image source stockée dans le plan FS1). Le résultat est chargé dans le plan FS0 de la carte Framestore.

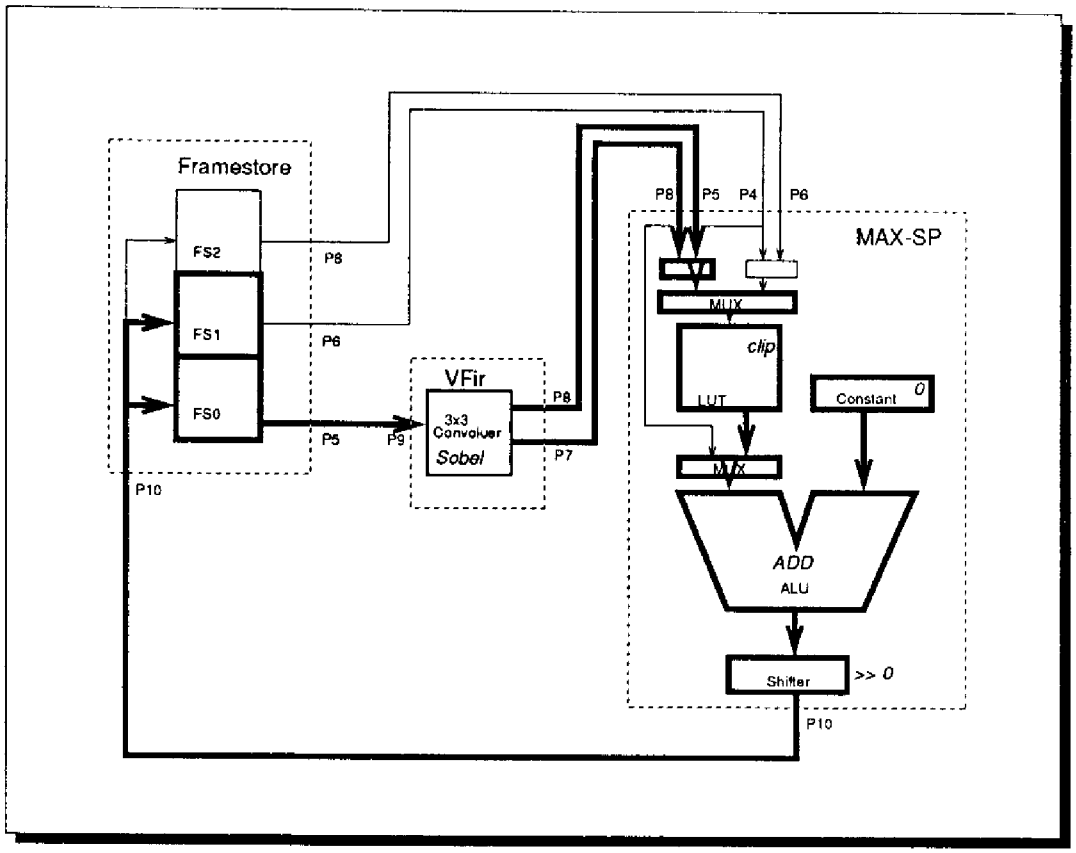


Figure A.2: *Calcul d'une convolution sur système Datacube : les éléments actifs durant ce calcul sont marqués en traits gras*

A.1.2 Calcul de la convolution

Pour réaliser la convolution, il est nécessaire de placer la carte FRAMESTORE en mode vidéo non entrelacé, transmettant une trame complète toutes les 40 millisecondes. L'opération de convolution est effectuée deux fois, avec les deux masques pour le calcul des dérivées en x et en y de l'image qui sont stockées dans les plans FS1 et FS2 de FRAMESTORE.

La convolution de l'image avec les deux masques du gradient de Sobel est réalisée sur la carte VFIR. Le résultat, calculé sur 12 bits, doit ensuite être ramené sur 8 bits. Pour cela nous utilisons une table de conversion de la carte MAX-SP qui réalise une fonction de coupure : les valeurs dont l'amplitude dépasse 127 sont limitées à l'intervalle $[-128 \ldots +127]$: les valeurs inférieures à -128 sont ramenées à -128, et celles supérieures à 127 sont ramenées à 127.

A.2 Tabulation du module et de la phase

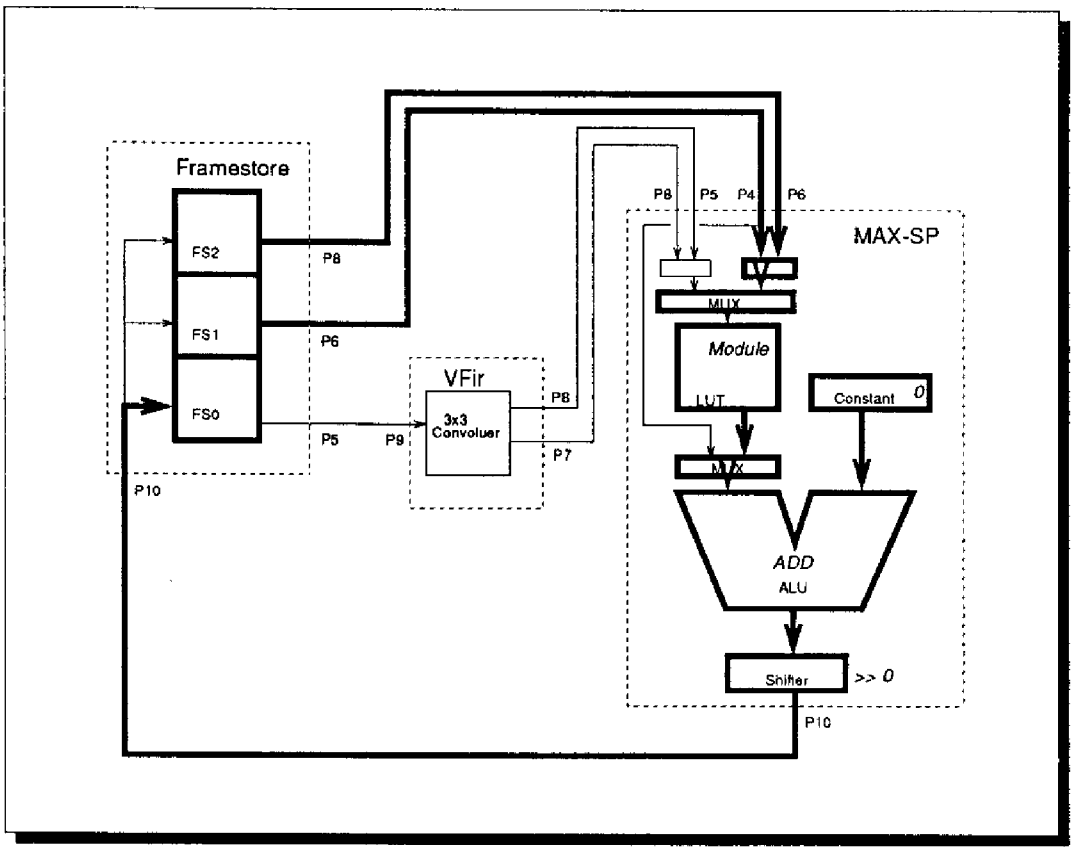


Figure A.3: *Calcul tabulé du module et de la phase*

A partir des dérivées en x et en y de l'image, le module et la phase sont calculés par des tables à l'aide de la carte MAX-SP.

Les dérivées en x et en y, tronquées de leurs deux bits de plus faible poids, sont dirigées vers les deux entrées de la LUT. La sortie est stockée dans le plan FS0 de FRAMESTORE. Les 12 bits d'adresse de la LUT sont vus comme une matrice carrée de 64 par 64 valeurs qui représentent les valeurs de -32 a +31 en complément à 2 (sur 6 bits) des gradients en X et en Y. La sortie de la LUT donne la valeur du module (resp. de la phase) du gradient pour les valeurs présentées en entrée. Après sauvegarde du plan FS0, l'opération peut être répétée en chargeant dans MAX-SP, la LUT permettant de calculer la phase.

A.3 Résultats

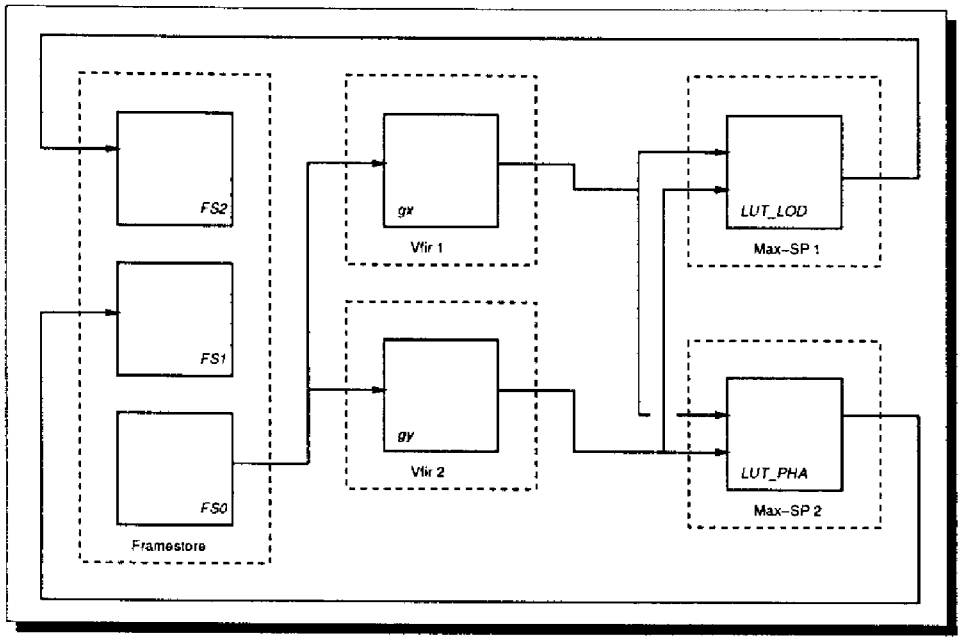


Figure A.4: *Calcul du gradient en pipe-line vrai en utilisant deux cartes VFIR et deux cartes MAX-SP*

En théorie, le traitement complet d'une image (calcul du module et de la phase) demande au minimum cinq temps de trame video en raison des nombreuses recirculations des images. Ceci permet donc une cadence maximale de cinq images par seconde. Malheureusement tous les chargements de paramètres sur les cartes empêchent d'utiliser pleinement toutes les trames, et en pratique plus d'une trame sur deux est perdue et la cadence atteinte n'est que de deux images par seconde.

Pour permettre un traitement qui soit effectivement temps réel vidéo, il faut éviter les recirculations de l'image pour réaliser un pipe-line véritable. Cela nécessite l'utilisation de deux cartes VFIR et deux cartes MAX-SP pour réaliser les chemins de données directs que l'on voit sur la figure A.4.

Glossaire

Bucket En français : baquet. Voir *h-coding*.

ERIM (caméra) Désigne la caméra laser 3D conçue au *Environmental Research Institute of Michigan*. Cette caméra se compose d'un télémètre laser à déphasage et d'un système de miroirs permettant de balayer un champ de 90° horizontalement et 40° verticalement en 0.5 seconde, échantillonnée en 256x64 pixels sur 8 bits.

Fil d'exécution (en anglais *execution thread*) : cette notion désigne un mode d'exécution d'une tâche dans un système d'exploitation qui utilise un pseudo-parallélisme interne sous forme de coroutines : plusieurs procédures s'exécutent en temps partagé au même instant à l'intérieur de la tâche en se partageant les données globales : ce sont les *fils d'exécution* de la tâche. Cette approche permet par exemple à une tâche de gérer simultanément plusieurs canaux de communication.

Focus of Expansion Point de fuite. Lors d'un mouvement en translation tous les mouvements apparents des points de l'image forment un faisceau de droites qui se coupent en un point : le *Focus of expansion (FOE)*.

H-coding Technique d'adressage des éléments d'un tableau qui fait intervenir un codage particulier de la clé de recherche. Un code est calculé pour chaque clé, et ces codes sont rangés dans une table spéciale la *h-table*, dont les éléments sont appelés *buckets* ou baquets. Pour rechercher un élément à partir d'une clé, on calcule le code associé. Une recherche dans la h-table fournit alors la liste des éléments correspondant à la clé.

Le cas le plus simple est celui où les clés sont numériques. La fonction de codage peut alors être l'identité échantillonnée sur un nombre réduit d'intervalles.

LUT *Look Up Table* Table de conversion. Les valeurs présentées en entrée servent d'indice dans une table qui donne la valeur de sortie. Très utilisées pour tabuler des calculs, mais aussi pour transformations ponctuelles dans une image ou les représentations en fausse couleur à partir de niveaux de gris.

Pixel *Picture Element* élément d'image. L'unité de numérisation spatiale de la fonction image. On dit aussi *PEL*.

Projection perspective Etant donné un point M de coordonnées $(X, Y, Z)^T$ dans l'espace à trois dimensions, sa projection perspective $m = (x, y)^T$ dans le plan xOy est définie par les équations :

$$x = X/Z \quad (1)$$

$$y = Y/Z \quad (2)$$

Par commodité, on réécrit souvent les équations 1 et 2 sous la forme (chapitres III et IV) :

$$\begin{pmatrix} \alpha x \\ \alpha y \\ \alpha \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Sténopé Le sténopé (du grec *stenos ôps*: petit trou) est un procédé de formation des images par un trou d'epingle percé dans une chambre noire. Il était utilisé par les peintres de la renaissance pour obtenir des images avec une perspective juste.

Références bibliographiques

[AFCET 89] 7ième Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle, Paris, (France), Décembre 1989.

[Aggarwal 90] J. K. Aggarwal & B. Sabata. *Motion from a sequence of range images.* In IEEE International Workshop on Intelligent Motion Control [IMCON 90], pages IP1–IP10.

[Ayache 88] N. Ayache. *Construction et Fusion de Représentations Visuelles 3D. Applications à la Robotique Mobile.* Thèse d'état, Université de Paris-sud, 1988.

[Ayache 89] N. Ayache. Vision stéréoscopique et perception multisensorielle. Applications à la robotique mobile. InterEditions, Paris, France, 1989.

[Basille 90] J.L. Basille. *How many dimensions for an array ?* In Proc. of Esprit BRA 3035 Workshop on Parallelism in image processing, pages 15–29, Chateau de Bonas, 32410 Castera-Verduzan, (France)., Août 1990.

[Bauzil 88] G. Bauzil, C. Lemaire & G. Vialaret. *Robot mobile HILARE, architecture et fonctionnalités de base.* Rapport technique 88132, Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Toulouse (France), 1988.

[Bockar 89] S. Bockar*et al. iWARP: An integrated solution to high-speed parallel computing.* Rapport technique CMU-CS-89-104, Carnegie Mellon University / Intel Corporation, Janvier 1989.

[Bolles 82] R.C. Bolles & R.A. Cain. *Recognising and locating partially visible objects : the local-feature-focus method.* International Journal of Robotics Research, vol. 1, no. 3, pages 57–82, 1982.

[Briot 90] M. Briot, M. Devy, J. Colly & L.H. Pampagnin. *Analyse des techniques de segmentation d'images 3D et application.* Rapport de recherche 90145, Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Mai 1990.

[Canny 83] J. Canny. Finding edges and lines in images. Master's thesis, MIT, Juin 1983.

[Casals 90] A. Casals, J. Amat & A. B. Martinez. *High speed processors for autonomous vehicles guidance.* In IEEE International Workshop on Intelligent Motion Control [IMCON 90], pages IP45–IP55.

[Chatila 90a] R. Chatila, M. Devy & M. Herrb. *Vision System Architecture for Environment Modelling and Motion Control of a Mobile Robot.* In A paraître. Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), 1990.

[Chatila 90b] R. Chatila & R. Ferraz De Camargo. *Open architecture design and inter-task/inter-module communication for an Autonomous mobile robot.* In IEEE International Workshop on Intelligent Robots and Systems [IROS 90].

[Chaumette 90] F. Chaumette. *La relation vision-commande : théorie et application à des tâches robotiques.* Thèse de l'Université de Rennes I, Institut de Recherches en Informatique et Systèmes Aleatoires (IRISA), Rennes (France), Juillet 1990.

[Chehikian 88] A. Chehikian, P. Stelmaszyk & S. De Paoli. *Hardware Evaluation process for tracking edge-lines.* In IAPR Workshop on Computer Vision, Special Issue on Hardware and Industrial Applications, Tokyo, Japan, Octobre 1988.

[Colly 89] J. Colly, M. Devy, M. Ghallab & L.H. Pampagnin. *Sensory Control for 3D Environment Modeling.* In IARP 1st Workshop on Multi-Sensor Fusion and Environment Modeling, Toulouse (France), Octobre 1989.

[Crowley 88] J.L. Crowley, P. Stelmaszyck & C. Discours. *Measuring Image Flow by Tracking Edge-Lines.* In Second International Conference on Computer Vision, Tampa, Florida (USA), pages 658–664, Décembre 1988.

[Crowley 90] J.L. Crowley, P. Bobet & K. Sarachik. *Dynamic World Modeling Using Vertical Lines Stereo.* In 1st European Conference on Computer Vision [ECCV 90], pages 241–246.

[Demazure 88] M. Demazure. *Sur deux problèmes de reconstruction.* Rapport de Recherche 882, Institut National de Recherche en Informatique et en Automatique (I.N.R.I.A.), Rocquencourt (France), Juillet 1988.

[Deriche 90] R. Deriche & O. Faugeras. *Tracking Line Segments.* In 1st European Conference on Computer Vision [ECCV 90], pages 259–268.

[Devy 90] M. Devy & J.M. Valade. *AMRZ Perception Subsystem — Architectural design.* Rapport technique eureka EU-18 (amr), Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Toulouse, France, Juin 1990.

[Dickmanns 90] E. D. Dickmanns. *Dynamic vision for intelligent motion control.* In IEEE International Workshop on Intelligent Motion Control [IMCON 90], pages IP67–IP74.

[Duclos 88] P. Duclos, F. Boeri, M. Auguin & G. Giraudon. *Image processing on a SIMD/SPMD architecture: OPSILA.* In 9th IEEE International Conference on Pattern Recognition [ICPR 88], pages 430-433.

[Duff 88] J. B. Duff, T. J. Fountain & K. N. Mattew. *The CLIP7A image processor.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, pages 310–319, 1988.

[ECCV 90] 1st European Conference on Computer Vision, Antibes (France), Avril 1990. Springer-Verlag.

[Faugeras 87a] O. D. Faugeras, F. Lustman & G. Toscani. *Motion and Structure from Motion from Point and Line Matches.* In Proc. ICCV'87, London (England), pages 27–35, 1987.

[Faugeras 87b] O.D. Faugeras & G. Toscani. *Camera calibration for 3D Computer Vision.* In Proc. International Workshop on Machine Vision and Machine Intelligence, Tokyo, Février 1987.

[Faugeras 89] O. D. Faugeras, N. Deriche & N. Navab. *From optical flow of lines to 3D motion and structure.* In Proc. IEEE International Workshop on Intelligent Robots and Systems, Tsukuba, Japan, pages 646-649, Septembre 1989.

[Ferraz 90] R. Ferraz De Camargo. *Hilare II: Architecture et Communications.* Rapport technique, Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Toulouse (France), Mai 1990.

[Fountain 86] T. J. Fountain. *Array architectures for iconic and symbolic image processing.* In 8th IEEE International Conference on Pattern Recognition, Paris (France), pages 24–33, Octobre 1986.

[Frau 90] J. Frau & V. Llario. *3D Tracking and adaptative motion prédiction of a target from a mobile robot.* In IEEE International Workshop on Intelligent Motion Control [IMCON 90], pages 433–437.

[Garnousset 86] H. Garnousset. *Interprétation du mouvement 3D: une approche par vision dynamique.* Thèse de troisième cycle, Université Paul Sabatier, Décembre 1986.

[Graefe 90] V. Graefe. *The BVV Family of robot vision systems.* In IEEE International Workshop on Intelligent Motion Control [IMCON 90], pages IP55–IP66.

[Grandjean 89a] P. Grandjean & A. Robert de Saint-Vincent. *3-D modeling of indoor scenes by fusion of noisy range and stereo data.* In IEEE International Conference on Robotics and Automation, Scottsdale, (USA), pages 681–687, Mai 1989.

[Grandjean 89b] P. Grandjean, A. Robert de Saint-Vincent & P. Moutarlier. *Calibration Odomètre - Platine Vision.* Note interne, Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Toulouse (France), 1989.

[Grandjean 90] P. Grandjean, M. Ghallab & E. Dekneuvel. *Multi-Sensory MOdel-Based OBject REcognition Through FUsion.* Rapport de recherche 90211, Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Juin 1990.

[Hamey 87] L. G. Hamey, J. A. Webb & I.C. Wu. *Low level vision on Warp and the apply programming model.* In J. Kowalick, editeur, Parallel Computation and Computers for Artificial Intelligence, pages 185–199. Kluwer Academic Publishers J., 1987.

[Hamey 89] L. G. Hamey, J. A. Webb & I.C. Wu. *An Architecture Independent Programming Language for Low-Level Vision.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 48, no. 2, pages 246–264, 1989.

[Heitz 89] F. Heitz & P. Bouthemy. *Estimation et segmentation du mouvement: approche bayésienne et modélisation markovienne des occlusions.* In 7ième Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle [AFCET 89], pages 1359–1368.

[Herrb 89] M. Herrb, F. Noreils & P. Grandjean. *Coordination perception-action en robotique mobile: poursuite d'un objet par vision 3D.* In 7ième Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle [AFCET 89], pages 1343-1358.

[Horn 81] B.K.P. Horn & B. G. Shunk. *Determining optical flow.* Artificial Intelligence, vol. 17, pages 185–203, 1981.

[Horswill 88] I. D. Horswill & R. A. Brooks. *Situated Vision in a Dynamic World: Chasing Objets.* In 7th National Conference on Artificial Intelligence (AAAI '88), Saint Paul (USA), Août 1988.

[ICPR 88] 9th IEEE International Conference on Pattern Recognition, Rome (Italy), Novembre 1988.

[IMCON 90] IEEE International Workshop on Intelligent Motion Control, Istanbul (Turkey), Août 1990.

[IROS 89] IEEE International Workshop on Intelligent Robots and Systems (IROS'89), Tsukuba (Japan), Septembre 1989.

[IROS 90] IEEE International Workshop on Intelligent Robots and Systems (IROS'90), Tsuchiura (Japan), Juin 1990.

[Jazwinski 70] A. Jazwinski. Stochastic Processes and Filtering Theory, volume 64 of *Mathematics in Science and Engineering*. Academic Press, 1970.

[Jolion 89] J.M. Jolion & A. Montanvert. *La pyramide adaptative: contruction et utulisation pour l'analyse de scènes 2D.* In 7ième Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle [AFCET 89], pages 197–206.

[Juvin 88] D. Juvin, J.L. Basille, H. Essafi & J.Y. Latil. *Sympati 2, A 1.5D processor array for image application.* In J.L. Lacoume, A. Chehikian, N.Martin & J. Malbos, editeurs, Signal Processing IV: Theories and Application, pages 311-314. Elsevier Science (North-Holland), 1988.

[Kubota 88] T. Kubota, H. Hashimoto & F. Harashima. *Path Searching of Mobile Robot Based on Cooperation of Sensors.* In IEEE International Workshop on Intelligent Robots and Systems, pages 569–574, Tokyo (Japan), 1988.

[Lee 90] S. Y. Lee & J. K. Aggarwal. *A System Design/Scheduling Strategy for Parallel Image Processing.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 2, pages 194–204, Février 1990.

[Little 87] J. J. Little, G. Blelloch & T. Cass. *Parallel Algorithms for computer vision on the Connection Machine.* In First International Conference on Computer Vision, London (Great Britain), pages 587–591, Juin 1987.

[Longuet-Higgins 81] H. C. Longuet-Higgins. *A computer algorithm for reconstructing a scene from two projections.* Nature, no. 293, pages 133-135, 1981.

[Matthies 89] L. Matthies. *Dynamic Stereo Vision.* PhD thesis, Carnegie Mellon University, Pittsburgh, Octobre 1989.

[Meer 89] P. Meer. *Stochastic image pyramids.* Computer Vision, Graphics, and Image Processing, vol. 45, no. 3, pages 269-294, Mars 1989.

[Merigot 89] A. Merigot, S. Bouaziz, F. Devos, J. Mehat, N. Y, P. Clermont & M. Ecrher. *SPHINX, un processeur pyramidal massivement parallèle pour la vision artificielle.* In 7ième Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle [AFCET 89], pages 185-196.

[Moutarlier 89a] P. Moutarlier & R. G. Chatila. *An Experimental System for Incremental Environment Modelling by an Autonomous Mobile Robot.* In Proc. ISER, Montreal, Juin 1989.

[Moutarlier 89b] P. Moutarlier & R. G. Chatila. *Stochastic Multisensory Data Fusion for Mobile Robot Location and Environment Modelling.* In Proc. International Symposium on Robotics Research, Tokyo, 1989.

[Moutarlier 89c] P. Moutarlier, P. Grandjean & R. Chatila. *Multisensory Data Fusion for Mobile Robot Location and 3D Modeling.* In IARP 1st Workshop on Multi-Sensor Fusion and Environment Modeling, Toulouse (France), Octobre 1989.

[Nagel 86] H. H. Nagel. *Images Sequences - Ten (octal) Years from Phenomenology towards a Theoretical Foundation.* In 8th International Conference on Pattern Recognition, Paris (France), Octobre 1986.

[Noreils 89a] F. Noreils. *Contrôle d'Exécution de Plans d'Actions et Architecture pour Robots Mobiles.* Thèse de l'Université Paul Sabatier, Toulouse (France) 560, Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Novembre 1989.

[Noreils 89b] F. R. Noreils & R. G. Chatila. *A General Structure for Mobile Robot Action Control.* In IEEE International Workshop on Intelligent Robots and Systems [IROS 89], pages 646–649.

[Noreils 90a] F. R. Noreils. *Integrating MultiRobot Cooperation in a Mobile-Robot Control System.* In IEEE International Workshop on Intelligent Robots and Systems [IROS 90].

[Noreils 90b] F. R. Noreils, M. Herrb & P. Grandjean. *A robust 3D vision module integrated in a mobile robot control system.* In IEEE International Workshop on Intelligent Motion Control [IMCON 90], pages 425–432.

[Orteu 90] J. J. Orteu, M. Devy & M. Briot. *Application of computer vision to automatic selective cutting with a roadheader in a potash mine.* Soumis à International Conference on Advanced Robotics (ICAR), Pise (Italia), 1990.

[Pampagnin 90] L.H. Pampagnin. *Reconnaissance d'objets tridimensionnels en perception monoculaire et multisensorielle — Application à la robotique spatiale —.* Thèse de l'Université Paul Sabatier, Toulouse (France) 779, Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Octobre 1990.

[Pham 89] H.N. Pham, J.N. Soulier, R. Pesty, P. Stelmaszyck, A. Chehikian & J.L. Crowley. *Cooperation between Stereovision and a token tracker process for the 3D vision of a mobile robot.* In IEEE International Workshop on Intelligent Robots and Systems [IROS 89], pages 171–177.

[Ranganathan 88] N. Ranganathan & M. Shah. *A Scale-Space Chip.* In 9th IEEE International Conference on Pattern Recognition [ICPR 88], pages 420–424.

[Robert 86] Arnaud Robert de Saint Vincent. *Perception et modélisation de l'environnement d'un robot mobile: une approche par stéréovision.* Thèse de l'Université Paul Sabatier, Toulouse (France), Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Novembre 1986.

[Sheen 88] A. Sheen. *Vision node Hardware architecture.* Esprit Project P1560 (SKIDS) Technical note TN 3103-01, SKIDS, Novembre 1988.

[Shen 90] J. Shen, S. Castan & J. Zhao. *A new passive measurement method by trinocular stereo vision.* Industrial Technology, vol. 1, no. 3, pages 231–259, 1990.

[Siahmed 86] N. Siahmed. *Contribution à l'intégration de la vision dynamique au système de perception 3D du robot mobile HILARE.* Thèse de docteur ingénieur, Université Paul Sabatier, Décembre 1986.

[Skordas 88] T. Skordas. *Mise en correspondance et reconstruction stéréo utilisant une description structurelle des images.* Thèse de docteur ingénieur, Institut National Polytechnique de Grenoble, Mars 1988.

[Tsai 87] R.Y. Tsai. *A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses.* IEEE Journal of Robotics and Automation, vol. 3, no. 4, pages 323–344, Août 1987.

[Tsuji 85] S. Tsuji, Y. Yagi & M. Asada. *Dynamic Scene Analysis for A Mobile Robot in a Man-Made Environment.* In IEEE International Conference on Robotics and Automation, St Louis (USA), pages 850–855, 1985.

[Tsuji 88] S. Tsuji. *Continuous image interpretation by a moving viewer.* In 9th IEEE International Conference on Pattern Recognition [ICPR 88], pages 514–519.

[Zavidovique 88] B. Zavidovique & E. Allart. *Functional computer for low level image processing.* In 9th IEEE International Conference on Pattern Recognition [ICPR 88], pages 830–832.

Thèse de Matthieu HERRB

Vision en mouvement pour la robotique mobile

Résumé :

Cette thèse traite le problème de l'utilisation de la vision artificielle en robotique mobile. La contribution présentée est orientée selon quatre axes : tout d'abord, les structures matérielles adaptées à l'exécution d'algorithmes de vision bas-niveau ont été étudiées : cartes spécialisées Datacube et machines parallèles à réseau de Transputers.

Le second axe traite du suivi et de la localisation d'objets tridimensionnels dans une séquence d'images, basé sur une prédiction au premier ordre du mouvement dans le plan des images et une vérification par recherche de cliques maximales dans le graphe des appariements compatibles. La localisation repose sur les sommets de l'objet et nécessite au moins cinq sommets.

Un module de modélisation dynamique de l'environnement utilisant la fusion numérique entre un module de stéréovision trinoculaire et un module de suivi des primitives appariées est présenté ensuite. Il permet d'acquérir des données tridimensionnelles robustes en fusionnant les observations 2D successives d'un segment 3D issu de la stéréovision.

Enfin, le dernier axe concerne l'intégration d'un tel système de perception dans l'architecture de contrôle d'un robot mobile pour réaliser différentes fonctions : mouvements asservis sur la vision, modélisation de l'environnement. Les modules fonctionnels implémentant la vision ainsi que les informations qu'ils échangent avec les autres modules du robot sont mis en évidence, ainsi que la manière dont ces modules interagissent avec la structure de contrôle du robot.

Différentes expérimentations réalisées sur le robot Hilare 1.5 ont permis de valider les algorithmes et les concepts proposés.

Mots clés : *Vision artificielle 3D, suivi de segments, stéréovision trinoculaire, robots mobiles, architecture de contrôle, calculateurs spécialisées et parallèles.*

Vision during Motion for Mobile Robots

Abstract :

This thesis deals with the problem of using computer vision in mobile robots. The contribution presented here is oriented along four axes : firstly, hardware architectures suited for the execution of low-level image processing algorithms are studied: the Datacube specialized cards and a parallel machine using a Transputer network.

The second axis deals with the tracking and the localization of a tridimensional object in a sequence of images, using first order prediction of the motion in the image plane and verification by a maximal clique search in the graph of mutually compatible matchings. The localization uses the object's vertices and requires that at least five of them be available.

A dynamic environment modeling module, using numerical fusion between trinocular stereovision and tracking of stereo-matched primitives, is than presented. It allows robust tridimensional data to be acquired by fusing successive 2D observations of a 3D edge given by the stereovision.

The final axis concerns the integration of this perception system in the control architecture of a mobile robot, to achieve various functions : vision servoed motion, and environment modeling. The functional units implementing vision tasks and the data exchanged with other units are exhibited, as well as the way in which they interact with the control structure of the robot.

Some experiments, realized with the mobile robot Hilare 1.5, have validated the proposed algorithms and concepts.

Keywords : *Computer 3D vision, edge tracking, trinocular stereovision, mobile robots, control architecture, specialized and parallel hardware.*